VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace SIP softwarového klienta pro platformu Android

Implementation of SIP Software Client for Android Platform

2014

Bc. Michal Oršel

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Michal Oršel**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Implementace SIP softwarového klienta pro platformu Android
Implementation of SIP Software Client for Android Platform

Zásady pro vypracování:

V současné době se rozšiřuje nabídka chytrých telefonů s OS Android. Na této platformě existují volně dostupné SIP aplikace, které však z velké části nepodporují režim šifrování SIP signalizace (SIPS), či šifrování RTP (SRTP případně ZRTP). Cílem práce je proto navrhnout a implementovat graficky a uživatelsky přívětivý SIP klient s podporou šifrování SIP i RTP pro platformu Android.

1. Studijní část: protokol SIP a jeho bezpečnost.
2. Teoretická analýza stávajících SIP řešení pod OS Android se zaměřením na zabezpečení komunikace.
3. Praktická implementace VoIP SIP klienta pod OS Android s podporou šifrování signalizace i RTP protokolu.
4. Testování a optimalizace vytvořené aplikace.
5. Vytvoření programátorské a uživatelské dokumentace k dané aplikaci.

Seznam doporučené odborné literatury:

[1] SIP Security by Dorgham Sisalem, John Floroiu, Jiri Kuthan, Ulrich Abend, Henning Schulzrinne (May 26, 2009)
[2] Hacking Exposed VoIP: Voice Over IP Security Secrets & Solutions by David Endler and Mark Collier (Nov 28, 2006).
[3] Android Forensics: Investigation, Analysis and Mobile Security for Google Android by Andrew Hoog (Jun 29, 2011)
[4] Android Apps Security by Sheran Gunasekera (Feb 24, 2012)

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Filip Řezáč**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 1. května 2014

Michal Ošel

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

Cílem této diplomové práce je prozkoumat a zhodnotit stav dostupných SIP klientů, především s ohledem na úroveň zabezpečení, a následně implementovat vlastní řešení. Nově vytvořený klient je napsán v jazyce Java a pro platformu Android, přičemž jako základ je použit framework PJSIP. Klient obsahuje správu uživatelských účtů, kontaktů, umožňuje hlasovou komunikaci a podporuje šifrování jak signalizace, tak datového toku.

Klíčová slova: Android, Bezpečnost, Java, Klient, PJSIP, RTP, SIP, Šifrování, TLS, VoIP

Abstract

The aim of the thesis is to explore and evaluate quality of available SIP clients, particularly with regard to level of security, followed by implementation of my own solution. Newly created client is written in language Java, with PJSIP framework used as a base, and is destined for Android platform. The client contains user accounts and contacts management, allows voice communication and supports both signalization and data stream encryption.

Keywords: Android, Client, Encryption, Java, PJSIP, RTP, Security, SIP, TLS, VoIP

Seznam použitých zkratek a symbolů

ADT	–	Android Development Tools
AES	–	Advanced Encryption Standard
DEA	–	Data Encryption Algorithm
DES	–	Data Encryption Standard
HMAC	–	Hash Message Authentication Code
HTTP	–	Hypertext Transfer Protocol
IDE	–	Integrated Development Environment
IETF	–	Internet Engineering Task Force
IPSec	–	IP security
ITU	–	International Telecommunication Union
JNI	–	Java Native Interface
MD5	–	Message-Digest 5
MIME	–	Multipurpose Internet Mail Extensions
NAT	–	Network Address Translation
NDK	–	Native Development Kit
OS	–	Operační Systém
PCM	–	Pulse-code modulation
PKI	–	Public Key Infrastructure
PSTN	–	Public Switched Telephone Network
RC4	–	Rivest Cipher 4
RFC	–	Request for Comments
RS	–	Retained Secrets
RSA	–	Rivest, Shamir, Adleman
RTP	–	Real-time Transport Protocol
SAS	–	Short Authentication String
SDK	–	Software Development Kit
SDP	–	Session Description Protocol
SHA1	–	Secure Hash Algorithm 1
SIP	–	Session Initiation Protocol
TCP	–	Transmission Control Protocol
TLS	–	Transport Layer Security
UAC	–	User Agent Client
UAS	–	User Agent Server
UDP	–	User Datagram Protocol
VoIP	–	Voice over Internet Protocol

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

S příchodem chytrých mobilních zařízení se rozšířila možnost využívat ke komunikaci vedle klasické telefonie i volání přes IP sítě. Samotná technologie VoIP vznikla již v 90. letech 20. století a je dodnes udržována a vyvíjena. S příchodem vysokorychlostní mobilních sítí a dobré dostupnosti připojení k internetu je využitelnost VoIP stále vyšší.

Od vzniku samotné myšlenky VoIP byl zahájen vývoj mnoha softwarových i hardwarových řešení, umožňujících internetovou telefonii. Výkon mobilních zařízení narostl a rovněž se rozšiřuje i počet aplikací, umožňující hlasovou i textovou komunikaci. Takové aplikace mohou využívat signalizační protokol SIP, neboť je velmi oblíbený, a to pro svou jednoduchost a snadnou implementovatelnost. Samotnou kapitolou jsou pak aplikace určené primárně pro mobilní telefony, které přímo konkurují klasické telefonii.

Úvod této práce je věnován právě rodině protokolů, které internetovou telefonii umožňují. Pro signalizaci (navazování hovorů, posílání zpráv) se dnes hojně využívá protokol SIP v kombinaci s protokolem SDP, který umožňuje vyjednat parametry spojení. Samotné spojení je pak realizováno protokolem RTP. V současnosti je možné všechny tyto protokoly šifrovat z důvodu ochrany soukromí. Důvodem vzniku této práce je právě stále narůstající počet útoků a absence kvalitního software, který by svými bezpečnostními prvky uživatele chránil. Právě na bezpečnostní stránku VoIP je v této práci kladen velký důraz.

Dále je v rámci této práce prostudován a zhodnocen aktuální stav dostupných SIP aplikací pro mobilní operační systém Android. Aplikace jsou hodnoceny z pohledu návrhu, použitelnosti, uživatelské přívětivosti a především pak bezpečnosti.

Po prostudování a zhodnocení všech potřebných informací je započat vývoj vlastního SIP klienta. Nejdříve je zvolen vhodný programovací jazyk, prostředí a potřebné technologie. Po navržení technické i vizuální stránky programu následuje jeho implementace.

Závěr práce je pak věnován zhodnocení vzniklého klienta, jeho dalšímu možnému rozvoji a případnému vylepšování.

2 Session Initiation Protocol

Protokol SIP je vyvíjen od roku 1996 pracovní skupinou Multiparty Multimedia Session Control v rámci **IETF**. Jako standard byl navržen (RFC 2543 [1]) v roce 1999. V roce 2002 byl SIP přepracován a popsán v novém standardu **RFC 3261** [2]. Protokol pracuje na aplikační vrstvě, patří do rodiny **VoIP** telefonie. Umožňuje navázání, ukončení a práci se spojením mezi dvěma nebo více účastníky sezení. Většinou se používá v kombinaci s protokoly **SDP** a **RTP** (více v kapitole 3).

Od myšlenky klasické telefonie (PSTN) se liší především svou end-to-end orientací. Zařízení používající SIP tedy udržují veškerou logiku, zatímco běžná telefonie používá primitivní koncová zařízení a logika je udržována v síti.

2.1 Prvky SIP sítě

Protokol SIP definuje několik typů prvků SIP sítě. K těm základním patří:

- User Agents
- Servery (proxy, registrar, redirect)

Koncové body SIP sítě jsou nazývány **User Agents** (UA). Mohou jimi být fyzické SIP telefony, aplikace nebo PSTN brány (které umožňují zprostředkovat komunikaci mezi SIP a běžnou telefonní sítí). Vedle UA se dále v SIP sítích nachází SIP servery, na které se UA připojují. Serverů může být několik typů a je obvyklé, že jsou všechny typy provozovány na jednom zařízení [3].

User Agents se skládá ze dvou logických částí: User Agent Server (UAS) a User Agent Client (UAC).

- UAC - část vysílající požadavky a přijímající odpovědi
- UAS - část přijímající požadavky a odesílající odpovědi
- B2BUA - speciální typ UA sloužící k přemostění

Je-li UA volající (tzn. odesílá požadavky), chová se jako UAC. Skrze SIP proxy server je zaslán požadavek (metoda **INVITE**) na sestavení spojení. Následně UA čeká na odpověď od volaného UA, který v rámci relace zastává roli UAS. Jakmile obdrží požadavek, zašle odpověď. Pokud chce volaný UA probíhající hovor ukončit, zavěsí hovor tím, že odešle zpráva metody **BYE**. V tomto případě se zpráva odešle přímo volajícímu, který se chová jako UAS a volaný jako UAC [3].

Speciálním typem UA je **B2BUA**. Vkládá se do cesty mezi dvě vytvářená spojení. V rámci spojení se chová jako SIP server s rozsáhlými možnostmi. Není však výkonný jako klasický SIP proxy.

2.1.1 Servery

Vedle UA jsou v infrastruktuře SIP sítě také SIP servery. K těmto serverům se UA přihlašují a tím informují o své aktuální poloze (v které síti a na které adrese se nachází, a to včetně informace o tom, jak se do dané sítě dostat). Úkolem SIP proxy serverů je pak směrovat žádosti o spojení mezi účastníky komunikace. Servery dále mohou provádět autentizaci, zaznamenávání doby trvání hovoru, přesměrovávání či jiné doplňkové služby [3].

Technologie SIP definuje dva základní typy SIP proxy serverů:

- Bezestavový (stateless)
- Stavový (stateful)

Primitivnější variantě SIP proxy serveru se říká **bezstavová**. Neuchovává si žádné informace o probíhajících komunikacích (transakcích ani dialozích). Signalizační zprávy, které příjme, pouze předá dál. Bezstavový server je rychlejší než stavový, ale nedokáže zachytit replikaci zpráv, ani neumí včas detekovat vznik smyček. Také neumožňuje provozovat některé druhy služeb, jako například: větvení či přesměrování. Bezstavový SIP Proxy server se převážně používá pro vyvažování zátěže v síti.

Stavový SIP proxy server je typ serveru, který si udržuje důležité informace o transakci či dialogu, až do jejich ukončení. S ohledem na to, jaké drží informace, rozlišujeme stavové SIP proxy servery na dva druhy:

- Transakční
- Dialogové

Transakční typ uchovává informace, které se vážou k žádosti. Tyto informace si uchovává až do doby, dokud není žádost definitivně vyřízena. Naopak **dialogové** servery si drží stav každého dialogu, až do doby, dokud není ukončeno celé spojení.

Přiřazování zpráv do transakcí dává serveru přehled o tom, zda již byla zpráva přijata či nikoliv. Výhodou je detekce replikace zpráv nebo smyček v síti. To serveru umožňuje větvení nebo podmíněné přesměrování provozu.

Některé služby, nabízené SIP proxy servery, je možné provozovat i samostatně. Specifikace protokolu SIP proto zavádí několik dalších typů serverů [3]:

- Registrar server
- Redirect server
- Location server
- B2BUA v režimu SIP server

2.2 SIP zprávy

Protokol SIP je textový, využívá kódování UTF-8 a formát jeho zpráv vychází z protokolu HTTP. K rozlišení uživatelů se využívá, na rozdíl od běžné telefonie, adres SIP URI. Tato adresa se podobá e-mailové adrese a obecně má následující tvar:

```
sip:user:password@host:port;parameters?headers
```

Adresa začíná schématem (sip: nebo sips: u zabezpečené varianty) následovaným uživatelským jménem. Nepovinně může následovat oddělovač (dvojtečka) a heslo. Po znaku musí být uveden hostitel, což může být doménové jméno nebo IP adresa (IPv4 nebo IPv6). Ostatní části (*parameters* a *headers*) vychází z obecné specifikace adresy URI, popsané v RFC 3986 [4]. Konkrétními příklady SIP URI adres jsou:

```
sip:jarek12@sipserver.cz
sip:123@example.com
```

Při pokusu o doručení žádosti se hledá SIP proxy sever, který zná polohu cílového UA. Právě k tomu se využívá část SIP URI obsahující informace o hostiteli.

2.2.1 Struktura zpráv

Zpráva protokolu SIP se skládá ze tří hlavních částí:

- Hlavička
- Pole hlavičky
- Tělo

Řádky jsou vždy ukončeny sekvencí znaků CRLF. První řádek hlavičky se liší v závislosti na typu zprávy. Pokud jde o žádost (request), je tvar řádku následující:

```
METHOD sip-uri SIP/2.0
```

V případě odpovědi (response) vypadá první řádek následovně:

```
SIP/2.0 result-code result-text
```

Metoda určuje typ zprávy, na kterém jsou závislé podmínky výskytu polí hlavičky. Rozdělení zpráv na různé typy je popsáno v následujících kapitolách.

Pole hlavičky je složeno ze dvou částí (klíč, hodnota) oddělených znakem dvojtečky. Jedno pole zabírá zpravidla jeden řádek, je však možné rozdělit hodnotu pole na více řádků (takové řádky jsou odsazeny bílými znaky). Polí může být libovolný počet a jejích výčet je ukončen dvojím odřádkováním.

Vše co následuje poté se nazývá tělo zprávy a jeho formát je opět závislý na použité metodě a obsahu polí v hlavičce. Následuje ukázka hlavičky zprávy, zachycené v reálném provozu [3]:

```
REGISTER sip:example.org SIP/2.0
Via: SIP/2.0/UDP 192.168.0.1:47618;rport;branch=z9hG4bK000000001
Route: <sip:example.org;transport=udp;lr>
Max-Forwards: 70
From: <sip:franta@example.org>;tag=000000002
To: <sip:franta@example.org>
Call-ID: 000000003
CSeq: 4 REGISTER
User-Agent: CSipSimple_thunderg-10/r2225
Contact: <sip:franta@192.168.0.1:47618;ob>
Expires: 900
Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, INFO, SUBSCRIBE,
  NOTIFY, REFER, MESSAGE, OPTIONS
Authorization: Digest username="franta", realm="example.org",
  nonce="000000005", uri="sip:example.org",
  response="1071eb565a5fd76f6fd75b055d5f1518"
Contact: <sip:franta@10.10.10.10:23116;ob>;expires=0
Content-Length: 0
```

2.2.2 Metody

Požadavky je také možné rozdělit podle použité metody. Ty určují záměr požadavku na sestavení, úpravu nebo ukončení spojení. Povinnost výskytu konkrétních polí v SIP hlavičce je dána použitou metodou. SIP definuje následujících 6 základních metod [3]:

- Metoda **INVITE**
- Metoda **ACK**
- Metoda **BYE**
- Metoda **CANCEL**
- Metoda **REGISTER**
- Metoda **OPTIONS**

2.2.2.1 Metoda INVITE reprezentuje žádost o zahájení spojení. Metodou je rovněž možné upravit parametry probíhajícího spojení (tomuto postupu se říká re-INVITE).

2.2.2.2 Metoda ACK se odesílá volanému k potvrzení přijetí konečné odpovědi na žádost **INVITE**.

2.2.2.3 Metoda BYE slouží k ukončení relace. Pokud chce účastník ukončit spojení, učiní tak odesláním zprávy metody **BYE**.

2.2.2.4 Metoda CANCEL ruší nabídku o zahájení spojení zprostředkovanou žádostí s metodou **INVITE**. Je tak možné učinit před doručením konečného potvrzení od volané strany.

2.2.2.5 Metoda REGISTER slouží pro registraci uživatele na Registrar server. Používá se i pro aktualizaci nebo zrušení registrace.

2.2.2.6 Metoda OPTIONS označuje speciální druh zprávy. Má shodnou strukturu s metodou **INVITE** a primárně slouží k získání vlastností SIP zařízení. Používá se také k ověření, zda je registrovaný uživatel stále dostupný. Je-li zařízení ukryté za **NAT**, je možné pomocí periodického ověřování udržovat dostupnost zařízení ze strany serveru.

Mimo tyto základní metody byly postupem času přidávány do SIP protokolu další rozšiřující metody. Mezi významné (z pohledu uživatele) lze zařadit metodu **MESSAGE**, která umožňuje zasílání textových zpráv (tzv. instant messaging).

2.2.3 SIP odpovědi

V případě doručení požadavku musí být odeslána příslušná odpověď. Je tak nutné reagovat na všechny metody, kromě metody **ACK**. První řádek hlavičky, který se u požadavků liší, je již popsán v kapitole 2.2.1. V řádku je zapsán kód odpovědi (result-code) a slovní vyjádření (result-text). Odpovědi se řadí do šesti kategorií, podobně jako je tomu v protokolu **HTTP**. Kódy odpovědí jsou očíslovány vždy trojciferným číslem. Jejich rozsah je od 100 do 699. Každá kategorie má vlastní význam [2, 3]:

- 1xx - bezproblémový průběh
- 2xx - zakončeno s úspěchem
- 3xx - probíhá přesměrování
- 4xx - chybný požadavek klienta
- 5xx - chyba serveru
- 6xx - fatální chyba

Odpovědi s čísly **1xx** nejsou konečné. Informují odesílatele o zpracovávání jeho žádosti. Typickými příklady jsou odpovědi 100, se slovním vyjádřením *Trying* a 180 s *Ringing*.

Odpovědi začínající na **2xx** jsou konečné a označují úspěšné dokončení požadavku. Nejčastějším případem tohoto rozsahu je odpověď 200 *OK*.

Číselná řada odpovědí **3xx** se používá k přesměrování a je vždy konečná. Odpovědi tohoto typu mohou nést několik druhů informací: Novou polohu volaného, užití jiné služby či kontakt na jiný SIP server. Například odpověď s číslem 380 *Alternative Service*.

Prvním zástupcem negativních konečných odpovědí je řada **4xx**. Odpověď tohoto typu je odeslána, pokud je detekována chybná syntaxe přijaté SIP zprávy nebo informace v této zprávě nejsou platné. Například 400 *Bad Request*.

Chyby serveru, jejichž příčinou pravděpodobně není špatný syntax přijaté SIP zprávy, ani údaje v ní, jsou indikovány odesláním odpovědi z rozsahu **5xx**. Například 500 *Server Internal Error*.

Je-li zaslán požadavek, jehož splnění není možné realizovat z důvodu nastalých okolností (nedostupnost spojení, apod.), je odpověď z rozsahu **6xx**. Například 603 *Decline*.

Příklad celé hlavičky SIP zprávy s odpovědí může vypadat následovně [2, 3]:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.0.1:47618;rport=23116;
  branch=z9hG4bK000000001;received=10.10.10.10
From: <sip:medik@iptel.org>;tag=000000002
To: <sip:medik@iptel.org>;tag=222222222
Call-ID: 000000003
CSeq: 4 REGISTER
Server: ser (3.3.0-pre1 (i386/linux))
Warning: 392 10.20.30.40:5060 "Noisy feedback tells: pid=1 ..."
Contact: <sip:franta@192.168.0.1:47618;ob>;expires=180
Content-Length: 0
```

2.2.4 Transakce

Komunikace v SIP je rozdělena na transakce. Označení transakce definuje posloupnost zpráv od prvotního požadavku po přijetí odpovědi. Transakce může obsahovat několik dočasných i konečných odpovědí. Více konečných odpovědí je dosaženo například v transakci **INVITE** při větvení. Navázání hovoru, požadavkem **INVITE**, je typickým příkladem transakce (blíže popsán v kapitole 2.3.2). Samotnou transakcí je také ukončení hovoru.

Aby nedocházelo k záměně, má každá transakce svůj identifikátor. Obvykle začíná řetězcem **z9hG4bK**. Je to z důvodu odlišení od SIP verze 1.0. Tento identifikátor je obsažen ve všech SIP zprávách dané transakce. Je uložen v parametru **branch** pole **Via**.

Identifikátor transakcí je používán především statefull servery. Tyto servery si v paměti udržují aktuálně probíhající transakce a doplňují je o nově příchozí zprávy. Díky těmto informacím transakční statefull servery znovu nereagují na doručené zprávy, které již byly zařazeny do transakce a zpracovány [2, 3].

2.2.5 Dialog

Dialog je posloupnost transakcí, z nichž pouze jedna může být aktivní. Jedná se tedy o soubor SIP zpráv mezi dvěma zúčastněnými stranami, které mají vzájemnou spojitost. Dialog řeší logiku řazení a směrování zpráv.

Dialogy, na rozdíl od transakcí, nemají svůj identifikátor přímo v SIP zprávách. Jsou identifikovány pomocí polí **Call-ID**, parametru **tag** v poli **From** a parametru **tag** v poli **To**. Jsou-li tyto hodnoty stejné ve dvou zprávách, pak tyto zprávy patří do stejného dialogu. Pro řazení zpráv v dialogu slouží pole **CSeq** [3].

Protokol SIP nedefinuje jednoznačně, kdy přesně je dialog sestaven. Dialogy jsou však tvořeny pouze pozitivními odpověď'mi. Jiné typy odpovědí nevrací pole potřebná pro identifikaci dialogu. Rozlišují se proto dva druhy:

- Napůl sestavený dialog (early dialog)
- Potvrzený dialog (confirmed dialog)

Hovoříme-li o napůl sestaveném dialogu, je to takový dialog sestavený dočasnou odpovědí s kódem v rozsahu **101-199**.

Potvrzený dialog je sestaven pomocí konečné pozitivní odpovědi s kódy řady **2xx**.

2.3 Logika

V této kapitole je popsáno, jakým způsobem probíhá registrování účastníků, volaní a zasílání textových zpráv. Nejdřív je nutné ujasnit si, co musí signalizační protokol zajistit:

- Lokalizaci volaného
- Zjištění stavu volaného (zda nemá obsazeno, přesměrování)
- Zjištění možností volaného (kodeky, rychlost)
- Vlastní navázání spojení (s pomocí SDP a RTP)
- Řízení spojení (změna parametrů a ukončení)

2.3.1 Registrace

Prvním, co musí UA udělat, je registrovat se na vybraný SIP server. Teprve až poté je možné s uživatelem navázat komunikaci. Registrace UA na serveru je však časově omezená a musí se periodicky obnovovat. Naopak zrušení registrace před uplynutím platnosti je možné zasláním žádosti o registraci s nastavenou požadovanou dobou registrace na hodnotu 0.

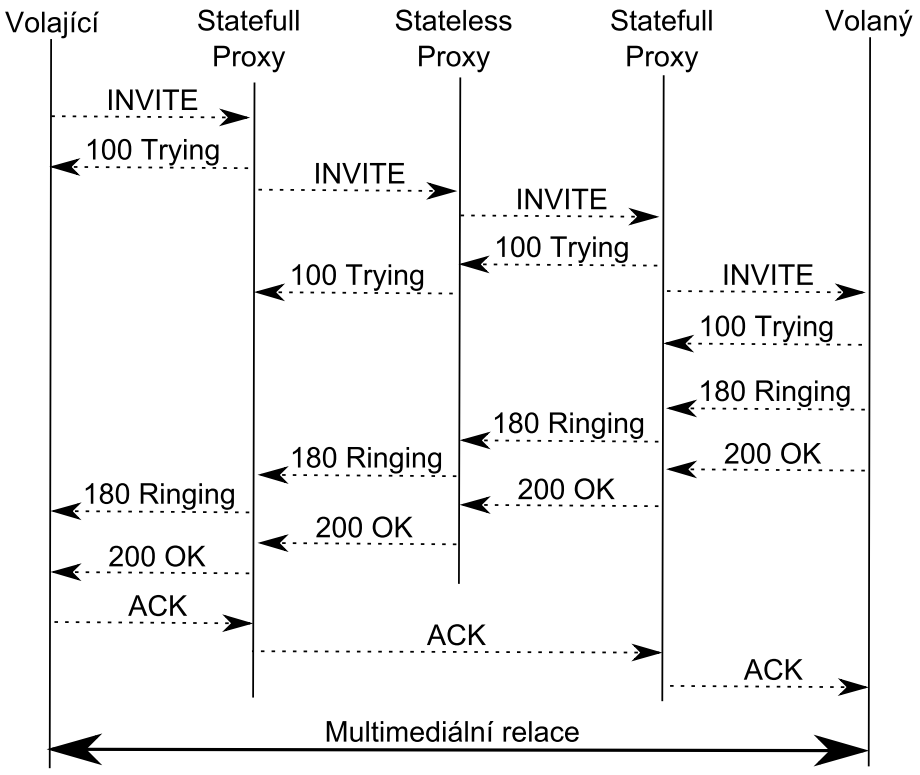
V hlavičce zprávy metody **REGISTER** jsou důležitá pole s názvem **From** a **Contact**. V poli **From** je uložena uživatelská URI adresa. V poli **Contact** je uvedena URI adresa zařízení. Registrační server si ukládá hodnoty a jejich vzájemné vazby. Klient v parametru **expires** pole **Contact** uvádí požadovanou dobu registrace. Není-li požadovaná doba v platném rozsahu, s ohledem na nastavení serveru, je serverem přenastavena na výchozí hodnotu [2, 3].

Při registraci může být serverem požadována autentizace uživatele, například metodou **HTTP Digest** (popsano v kapitole 2.4). První žádost o registraci je zaslána bez

jakýchkoliv autentizačních údajů. Server ji zamítne a jako odpověď zašle 401 *Unauhorized*. Do odpovědi přidá pole **WWW-Autenticate** s hodnotami **realm** a **nonce**, potřebnými k autentizaci. Uživatel zašle žádost o registraci znovu, ovšem s nově přidaným polem **Authorization**. Toto pole obsahuje veškeré potřebné autentizační údaje, vypočtené na základě požadavků serveru. Hodnota obsažená v parametru **response** je výsledkem hashovací funkce **HMAC-MD5**, do které bylo vloženo uživatelské jméno, hodnota *realm*, hodnota *nonce* a heslo uživatele [2, 3].

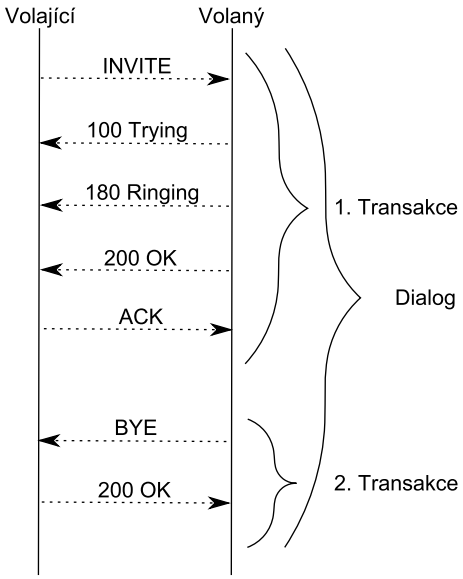
2.3.2 Hovor

Hovor je multimediální relace navázaná mezi dvěma nebo více účastníky. Iniciátor hovoru odešle volanému (volaným) zprávu s žádostí metody **INVITE**. Pokud druhá strana je schopná požadavek přijmout, odešle dočasnou odpověď (100 *Trying* a poté 180 *Ringing*). Dočasnou odpověď s kódem 100 může odeslat nazpět i statefull proxy server. Pokud volaný účastník hovor příjme, vrátí konečnou odpověď s kódem 200 a volající potvrdí přijetí odesláním nové zprávy metody **ACK**, po které je navázána multimediální relace. Odmítne-li účastník hovor přijmout, zašle volajícímu zprávu metody **BYE**. Vše je blíže znázorněno v obrázku 1 [2, 3].



Obrázek 1: Schéma hovoru protokolu SIP [3]

Rozčlenění hovoru na dialogy a transakce popisuje obrázek 2.



Obrázek 2: Rozlišení dialogu a transakce během hovoru protokolu SIP [3]

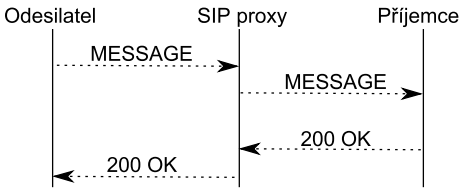
2.3.3 Textové zprávy

Jedno z rozšíření protokolu SIP (**RFC 3428** [5]) umožňuje zasílání textových zpráv. Tyto zpravy se spíše než SMS zprávám, známých z protokolu GSM, podobají spíše internetové diskuzi s potvrzením o doručení. Mohou dosahovat větší délky než zmíněné SMS a lépe zpracovávají znaky s diakritikou.

Zpráva je odeslána pomocí žádosti **MESSAGE**. Text samotného sdělení je obsažen v těle SIP zprávy. Obsah zprávy může být různého typu. Ten, stejně jako u emailových zpráv, je určen pomocí technologie **MIME**. Pro běžné zprávy se v hlavičce nachází pole:

```
Content-Type: text/plain
```

Příjemce zprávy (UAS) po doručení žádosti **MESSAGE** ihned odešle nazpět zprávu s konečnou odpovědí. V případě, že zprávu úspěšně přijal (tato informace nekoresponduje s tím, zda uživatel zprávu viděl nebo byla zobrazena), odešle odpověď s kódem 200 *OK*. Nebyla-li zpráva úspěšně doručena, odešle odpověď **4xx** nebo **5xx**. V případě odmítnutí zprávy pak **6xx** [3].



Obrázek 3: Schéma zobrazující odeslání textové zprávy [3]

Celá délka zprávy (včetně hlavičky) nesmí překročit 1300 bajtů. V případě delší zprávy je nutné obsahovou část předat v rámci dialogu, popsaného protokolem **SDP**. Základní zpráva **MESSAGE** ale žádný dialog nenavazuje, jedná se pouze o transakci.

2.4 Bezpečnostní rizika

Protokol SIP ke svému fungovaní využívá mnoho dalších protokolů. Zranitelnost každého z nich přímo či nepřímo ovlivňuje jeho bezpečnost. V této kapitole bude popsáno, jaká bezpečnostní rizika se skrývají v samotném SIP protokolu a jeho mechanismech.

Rizikům VoIP telefonie se věnuje organizace Voice over IP Security Alliance (VOIPSA, Inc.), od roku 2005. Tato organizace si klade za úkol zlepšit bezpečnost VoIP komunikace a řadí možná rizika do několika skupin [6]:

- **Sociální** - obtěžování, narušování soukromí
- **Odposlouchávání** - skenováním provozu
- **Zachycení a úprava** - přesměrování, změna identity
- **Zneužití služby** - podvody s registrací, obcházení autentizace, krádež a zneužití účtu
- **DoS útoky** - (Denial of Service) zahlcení a tím způsobení nedostupnosti služby
- **Vyčerpání zdrojů** - napájení, navýšení latence

Z těchto útoků se pouze některé týkají samotného SIP klienta, ostatní jsou útoky na serverovou část, která v této práci není zohledněna.

2.4.1 Sociální typ útoku

Sociálním typem útoku se rozumí takový útok, který obtěžuje či jinak narušuje soukromí. Pro tento typ útoku není nutné používat speciální software.

Nejméně ohrožujícím zástupcem této skupiny, z pohledu dopadu na bezpečnost, je nevyžádané reklamní sdělení (Spam). Naopak mezi ty závažné patří útok zvaný *phishing*, kdy útočník vystupuje pod cizí identitou a uvede tak uživatele v omyl. Uživatel může nevědomky prozradit citlivé informace (například hesla nebo kontaktní údaje) a snadno se může stát obětí trestného činu (podvodně uzavřené smlouvy, vydírání apod.) [6, 7].

2.4.2 Odposlouchávání

Odposlouchávání vychází z absence šifrování komunikace a spočívá v zachycování datového toku a jeho rekonstrukce. Útočník je tedy schopen zachytit celý hovor a získat tak citlivé informace.

Tento typ útoku se nevztahuje pouze na přenášená data, ale i na informace přenášené signalizačním protokolem. Takto lze získat informace nutné například k autentizaci uživatelského účtu apod.

2.4.3 Úprava probíhajícího hovoru

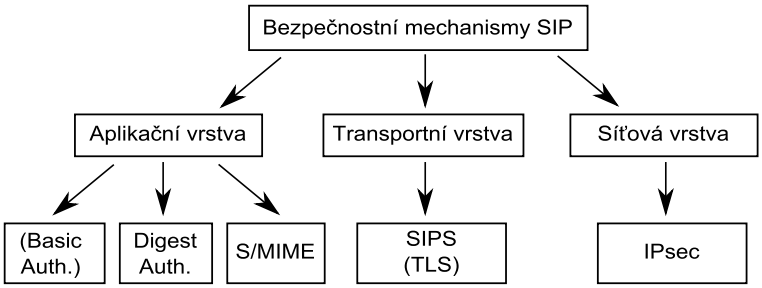
Dalším velmi závažným rizikem je úprava probíhajícího hovoru. Útočník, který zachytí SIP komunikaci, může změnit předávané údaje nebo dokonce provoz přesměrovat. Provoz může být přesměrován například na kontakt se speciálním tarifem a z něj opět zpátky, nebo naopak může útočník podvrhnout RTP paket. Rovněž je možné konkrétní úpravou dat v RTP paketu využít chyb v implementaci některých hardwarových prvků sítě [6, 7].

2.4.4 Zneužití služby

Tento typ útoku se týká převážně serverů, může se však vyskytovat v kombinaci s chybnou implementací klienta. Příkladem může být získání hesla z úložiště aplikace nebo možnost získání dat při nekorektní implementaci síťové části.

2.5 Zabezpečení signalizace

Obecně je v rámci zabezpečení nutno držet se několika základních bezpečnostních principů, které samy o sobě snižují možné bezpečnostní riziko. Samotný protokol SIP obsahuje několik bezpečnostních mechanismů, které jsou vysvětleny dále v této práci.



Obrázek 4: Rozdělení bezpečnostních mechanismů protokolu SIP [8]

2.5.1 Bezpečnostní principy

Aby bylo možné protokol označit za bezpečný, musí být zajištěno několik základních bezpečnostních principů. Jejich popis s vysvětlením je obsahem následujících odstavců [8].

Důvěrnost znamená, že nikdo jiný, kromě odesílatele a příjemce, nebude moci získat obsah přenášené informace. Pro zajištění důvěrnosti je tedy nutné informace zašifrovat.

Autentičnost zajišťuje ověření pravosti (původu) přijatých informací. Jedná se o ochranu před zasláním falešných a podvržených údajů.

Autorizace ověřuje, zda může uživatel provádět určité operace s ohledem na jemu přidělená oprávnění. Autorizovat je možné s ohledem na předchozí autentizaci.

Integrita umožňuje ověřit integritu informací, tzn. zda byla data na trase od odesílatele k příjemci někým modifikována.

2.5.2 HTTP Basic Authentication

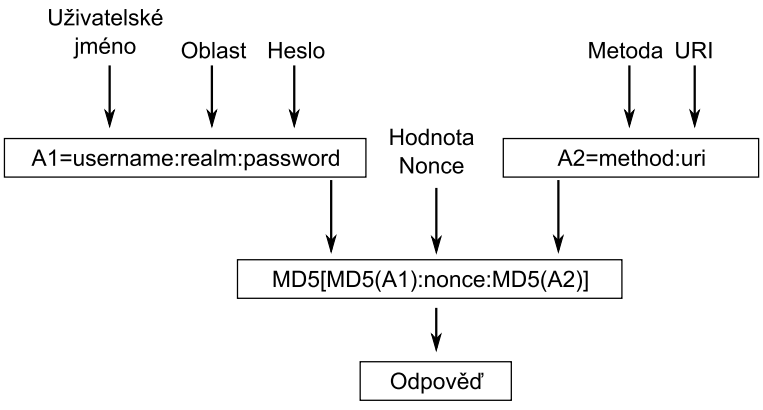
Tato základní autentizační metoda zajišťuje pouze ověření uživatele pomocí sdíleného hesla, zaslaného v nešifrované formě. Tento způsob zabezpečení není příliš vhodný a přímo specifikací protokolu SIP není doporučen [3, 9].

2.5.3 HTTP Digest Authentication

Jedná se pouze o rozšíření autentizační metody s tím rozdílem, že ověřovací heslo není zasláno v čisté formě. Je použita hashovací funkce **HMAC-MD5** či **HMAC-SHA1**, která z řetězce složeného z uživatelského jména a hesla vypočte otisk. Tento otisk se poté zašle nešifrovaně, není z něj ale možné odvodit původní heslo [3, 9].

Samotná komunikace pak probíhá následovně: Pošle-li UAC požadavek metodu REGISTER na SIP server, je žádost zamítnuta odpovědí 407 *Proxy Authentication Required*. Odpověď také obsahuje informaci o tom, jaká autentizační metoda má být použita (v tomto případě Digest). Při opětovném odeslání požadavku (metody REGISTER) je již do zprávy klientem vloženo pole *WWW-Authenticate* se všemi autentizačními parametry. Mezi základní parametry patří:

- **Authentication Scheme** - hodnota *Digest*
- **Realm** - určuje rozsah platnosti ověření (většinou doména)
- **Nonce** - hodnota generována UA, který se autentizuje serveru



Obrázek 5: Způsob výpočtu odpovědi schéma Digest [8]

2.5.4 Secure MIME (S/MIME)

Jedna z možných metodik zabezpečení, která se také používá u elektronické pošty, se jmenuje S/MIME. Tato technologie umožňuje digitálně podepsat a šifrovat přenášená data, ne však hlavičku zprávy, což umožňuje síťovým prvkům s ní pracovat. Používá k tomu typ **application/pkcs7-mime**. Je možné použít i typ **multipart/signed**, který umožňuje přidat do obsahu zprávy i nešifrovaná data.

Autentizace zajišťuje architektura veřejných klíčů **PKI**. K ochraně přenášených dat se využívají symetrické kryptografické metody **DES**, **Triple DES** či **AES** [3, 9].

2.5.5 SIPS URI (TLS)

Pokročilé zabezpečení SIP signalizace nabízí technologie TLS, která se vloží mezi transportní a aplikační vrstvu. Pro identifikaci této metody se mění prefix hlavičky na **SIPS**. Uvedení prefixu zaručuje, že všichni účastnící spojení, kteří spravují tento požadavek, přenáší data zabezpečeně. Podmínkou této metody je použití protokolu TCP.

Samotné TLS je pak složeno z několika částí. Nejprve je nutné provést autentizaci a dohodnout komunikační parametry (šifrovací nebo hashovací algoritmus, symetrický klíč). Následně jsou přenášena již šifrovaná data.

K autentizaci se využívá architektury veřejných klíčů. Vždy se autentizuje server klientovi, volitelně je pak možno nastavit, aby server ověřoval klienta. Konkrétně UAC vyžaduje od UAS platný certifikát podepsaný důvěryhodnou certifikační autoritou. Na straně UAC je doporučeno mít nastavený tzv. kořenový certifikát, který je vydáván samotnou certifikační autoritou a je pomocí něj možné ověřit platnost certifikátu UAS [8].

Samotná data mohou být šifrována metodami **DES**, **Triple DES**, **AES**, **DEA** nebo **RC4**. Výměnu symetrických klíčů zajišťuje asymetrické šifrování **RSA** [3, 9].

Technologie TLS neposkytuje zabezpečení přenosu na celé komunikační trase, ale pouze s prvním SIP serverem. Ten příchozí data musí dešifrovat, aby získal informace o tom, kam zprávy dále směrovat. Následně server vytvoří další TLS spojení pro přenos směrem k volanému. Jedná se tedy o metodu hop-by-hop.

2.5.6 IP Security

Pro zabezpečení komunikace je také možné použít protokol IPsec. Ten umožňuje zabezpečení nejen signalizačních zpráv SIP protokolu, ale také multimediální data přenášena prostřednictvím protokolu **RTP** (či jeho modifikací). Technologie IPsec vytvoří bezpečný kanál před samotným započetím komunikace. Velkou výhodou IPsec je, že se zabezpečený kanál nachází na síťové vrstvě. Zabezpečuje tedy celé IP datagramy, což zaručuje nezávislost na protokolech vyšších vrstev **TCP**, **UDP** [9].

Obashuje dva protokoly: **Authentication Header Protocol** (AH), který zaručuje autenticitu a integritu dat včetně ochrany proti některým útokům, a **Encapsulating Security Payload Protocol** (ESP), který také garantuje důvěru nespojové komunikace a obsahuje jednoduchý mechanismus proti monitorování provozu. AH a ESP mohou být použity v kombinaci nebo každý jednotlivě [8].

3 Real-time Transport Protocol

S ohledem na nutný rozvoj telekomunikací v IP sítích byla organizací **IETF** zřízena pracovní skupina **Audio-Video Transport Working Group**, která si kladla za cíl vytvořit protokol pro doručování zvukových a obrazových dat v reálném čase.

Výsledkem jejich práce je protokol RTP (Real-time Transport Protocol), který je určen právě pro přenos multimediálních dat jako je například hlas či video. Poprvé byl zveřejněn v **RFC 1889** [10], vydaném roku 1996. V roce 2003 vyšlo **RFC 3550** [11], které přináší drobná vylepšení.

RTP je protokolem aplikační vrstvy, je nezávislý na typu sítě a přenosovém protokolu. V IP síti se nejčastěji používá v kombinaci s protokolem **UDP**, který na rozdíl od protokolu **TCP** negarantuje doručení paketu, tedy odpadá zpomalení způsobené kontrolní režií a řazením dat. U přenosu v reálném čase se ztracené nebo poškozené pakety ignorují, protože jejich rekonstrukce či znovuzaslání by bylo časově náročné a paket by do té doby pozbyl platnosti. RTP byl navržen jak pro individuální, tak pro skupinové přenosy dat v jednom i obou směrech přenosu [12].

V kombinaci s protokolem RTP se využívá protokol **RTCP**, který neslouží k přenosu samotných dat, ale poskytuje informace o navázaném RTP spojení, kvalitě přenosu, ztrátovosti a dalších statistických údajích [11].

RTP nemá vyhrazený port, ale typicky používaná čísla portů jsou: 5004, 5005, 6970.

RTP zajišťuje:

- Správné řazení paketů, je totiž možné, aby později odeslaná data dorazila dříve než dříve odeslaná.
- Časové značky, zajišťují přehrání ve správném čase.
- Typ přenosu a jeho kódování.

RTP nezajišťuje:

- Rezervaci kanálu.
- QoS

3.1 Struktura RTP paketu

Ve své podstatě je struktura protokolu jednoduchá. Každý datový paket obsahuje pouze hlavičku a data.

3.1.1 Popis hlavičky

Hlavička má velikost minimálně 12 bajtů.

Význam	Bity	Popis
Version	2	Určuje verzi RTP protokolu.
Padding	1	Existenci patičky na konci paketu. Poslední bajt patičky obsahuje její velikost v bajtech (včetně).
Extension	1	Existence rozšiřující hlavičky mezi záhlaví a daty.
CSRC count	4	Obsahuje počet CSRC za hlavičkou.
Marker	1	Označuje konec rámce v paketovém toku.
Payload Type	7	Určuje formát užitečného zatížení RTP (viz. 3.1.1).
Sequence Number	16	Číslování paketů. S každým odesláním RTP paketu je o jedna vyšší. Počáteční číslo určí vysílač. Slouží k správnému seřazení paketů a identifikaci chybějícího paketu.
Time stamp	32	Časová značka. Vyjadřuje vzorkovací značku prvního oktetu v RTP paketu. Značka musí být odvozena od lineárního časovače z důvodu synchronizace a korekce rozptylu zpoždění.
SSRC	32	Jednoznačný identifikátor synchronizačního zdroje vysílání. Hodnota je zvolena náhodně.
CSRC	32	Identifikační seznam přispívajících zdrojů dat.
Header Extension	32	Volitelné rozšíření hlavičky využívaně některými implementacemi (způsoby kódování nebo protokoly). První záznam obsahuje specifický identifikátor profilu 16 bitů a délku specifikace 16 bitů, která označuje, kolik 32 bitových jednotek následuje.

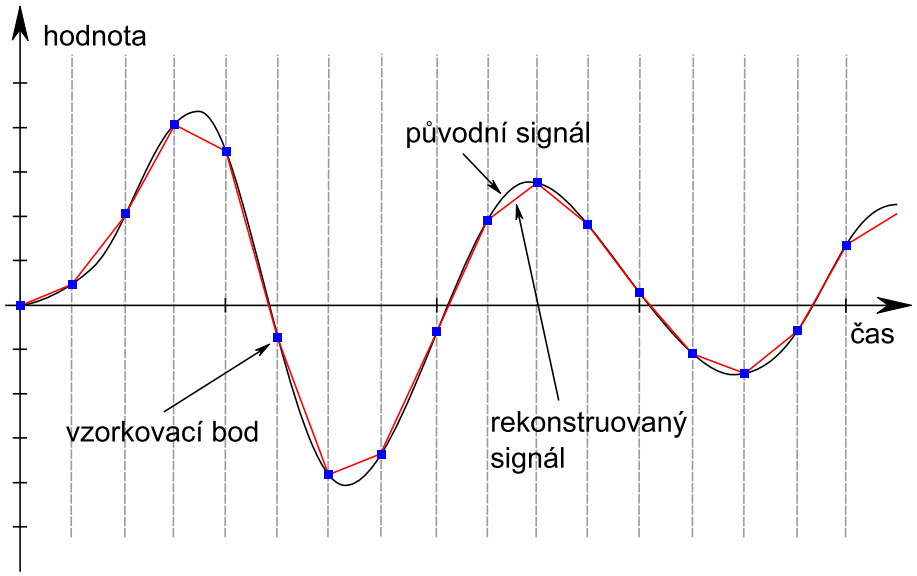
Tabulka 1: Struktura hlavičky RTP paketu [11]

Payload Type určuje formát užitečného zatížení RTP jako je typ přenášených dat, kodek, vzorkovací frekvence. Ve standardu jsou definovány například tyto H.263, H.264, JPEG, MPEG, G.711. Nepodporované formáty lze dohodnout dynamicky například za pomocí protokolu SDP před započetím samotného přenosu. Rozsah dynamických kódů je 96 - 127.

3.1.2 Princip činnosti RTP

Hlas vstupující do mikrofonu je digitalizován (obvyklá vzorkovací frekvence je 8KHz) za pomocí pulzně kódové modulace PCM (standard ITU G.711.), jak ukazuje obrázek 6, a zakódovám libovolným kodekem. Před tato zakódovaná zvuková data je postupně vložena RTP hlavička a poté UDP hlavička.

Samotné RTP pakety jsou odesílány v pravidelných intervalech. Pro přenos audia a videa, například při konferenčních videohovorech, musí být zřízeny dvě oddělené relace.



Obrázek 6: Způsob vzorkování zvukového signálu [13]

3.2 SRTP

Velkou slabinou protokolu RTP absence jakékoliv formy ochrany soukromí. Není proto problém, například za pomoci programu Wireshark, zachytit RTP spojení , následně zrekonstruovat a poté přehrát. Protokol RTP je také náchylný k vložení třetí strany do hovoru. Tím je myšleno možné přesměrování nebo vmísení se do hovoru útočníkem.

Rozšíření jménem SRTP (Secure Real-time Transport Protocol) vkládá do RTP bezpečnostní mechanismy. Zveřejněno bylo roku 2004 v **RFC 3711** [14]. Zmíněný předpis se také věnuje protokolu **RTCP** a definuje jeho zabezpečenou verzi **SRTCP** (Secure RTCP) [14].

3.2.1 Nová pole v SRTP paketu

- Master Key Identifier
- Authentication Tag

SRTP rozšiřuje původní RTP paket o dvě nová pole. Obě pole jsou délky 32 bitů. Nakonec paketu bylo přidáno pole **Master Key Identifier**, které je nepovinné. Následuje pole **Authentication Tag**, které je doporučené.

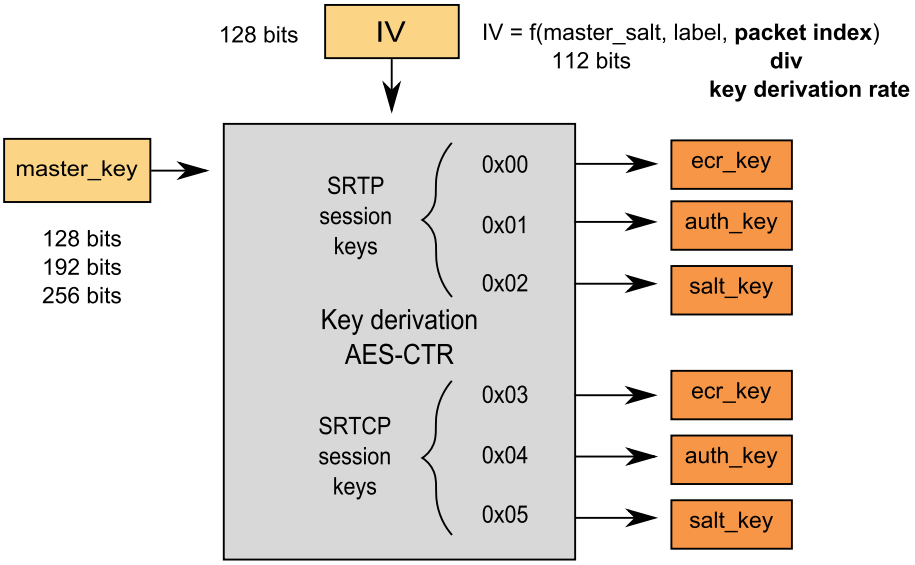
3.2.2 Nová pole v SRTCP paketu

- SRTCP index
- Master Key Identifier
- Authentication Tag

Do paketu bylo přidáno pole jménem **SRTCP index**, které slouží jako počítadlo paketů. Toto pole má i jinou funkci; první bit má speciální význam a je označen jako **Encription Flag**. Ten značí, zda bylo šifrováno tělo SRTCP paketu. Paket SRTCP je rozšířen taktéž o pole **Autenticaion Tag**, které je ale v tomto případě povinné [3].

3.2.3 Samotné zabezpečení

Nové pole **Master Key Identifier** identifikuje hlavní klíč tzv. master key. Na základě tohoto klíče jsou vygenerovány všechny tajné symetrické klíče sezení tzv. session keys. Při první komunikaci si strany vymění master key, který může mít 128, 192 či 256 bitů, většinou za pomocí protokolu **SDP**. Protokol **SDP** nepoužívá žádnou formu zabezpečení, hrozí proto riziko zachycení klíče útočníkem (například útokem známým jako Man-in-the-Middle) a následné prolomení komunikace [16].



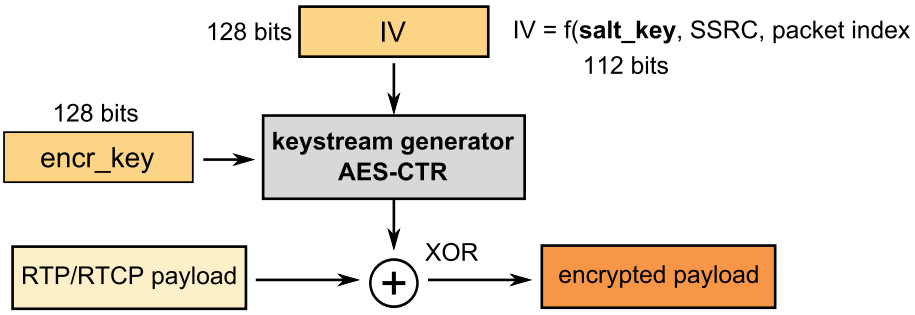
Obrázek 7: Způsob generování session klíčů z Master klíče [3]

SRTP i SRTCP zavádí následující bezpečnostní prvky:

- Důvěrnost informace
- Autentičnost zpráv
- Integritu

Důvěrnost informace zajišťuje v SRTP symetrická kryptografická metoda **AES-CTR**, která v tomto případě slouží jako generátor pseudonáhodných klíčů. Vstupem do generátoru je inicializační vektor IV, který se skládá z kontrolního součtu salt_key, SSRC

(náhodné číslo jednoznačně identifikující zdroj) a paket indexu. Dalším vstupem do generátoru je jeden z klíčů sezení a to šifrovací klíč (encryption key). Výsledný pseudonáhodný klíč je pomocí operace XOR aplikován na nezabezpečený obsah paketu [16].



Obrázek 8: Kódování obsahu paketu [3]

Autentičnost a integrita zpráv je zajištěna pomocí algoritmu **HMAC-SHA1**. Tímto algoritmem je vytvořen kontrolní součet z hlavičky a obsahu **SRTP** paketu (hlavička spolu s užitečnými daty a **RTP** paddingem). Délka kontrolního součtu, produkovaného metodou **HMAC-SHA1**, je 160 bitů. Ten je poté zkrácen na 80 nebo 32 bitů vložen do pole **Authentication Tag** [3].

3.3 ZRTP

ZRTP je protokol navržený Phillem Zimmermannem v roce 2006. Standardizován byl v roce 2011, **RFC 6189** [15]. Jedná se o rozšíření protokolu **SRTP** o Diffieho-Hellmanovu výměnu klíčů, metodu **SAS** a **RS**. ZRTP je odolné proti Man-in-the-Middle. Protokol ZRTP nahrazuje nezabezpečenou výměnu master klíče. Ten je v SRTP zasílán v nezabezpečeném SDP. Po zajištění bezpečné výměny klíčů se provede přepnutí do SRTP módu [16, 17].

Pořadí módů:

- Detekce, zda je přítomna podpora ZRTP.
- Výměna symetrických klíčů.
- Přepnutí do SRTP.

ZRTP definuje následující šifrovací metody:

- AES-128
- AES-192
- AES-256

3.3.1 Diffie-Hellmanův algoritmus

Tato technika umožňuje počáteční výměnu symetrických klíčů, bez předem dohodnutého šifrovacího klíče (předchozího sdíleného tajemství). Princip spočívá v tom, že si každá ze stran vygeneruje dva klíče, soukromý a veřejný. Veřejný klíč je poslán druhé straně, kde je z něj, za pomocí matematických operací, odvozen symetrický klíč [17].

Pro počáteční výměnu klíče ZRTP nabízí 2 módy Diffie-Hellmanova algoritmu:

- DH-3072
- DH-4096

3.3.2 Metoda SAS

Samotná technologie Diffieho-Hellmana neposkytuje ochranu proti útoku Man-in-the-Middle. Proto ZRTP v rámci tohoto protokolu používá technologii SAS. Metoda je použita k autentizaci druhé strany. SAS detekuje, zda není v průběhu vytváření relace cílem útoku Man-in-the-Middle. Hodnota SAS je vypočtena jako hash symetrického klíče dohodnutého metodou Diffeiho-Hellmana. Hash je vygenerován na obou komunikujících stranách a poté zaslán. Neshodují-li se klíče, je relace s velkou pravděpodobností předmětem útoku [15].

Použití Diffieho-Hellmana s technologií SAS omezuje útočníka na uhádnutí správné hodnoty v krátkém časovém intervalu. Čím větší je délka SAS, tím je šance na uhodnutí klíče menší. Například, je-li SAS o délce 16 bitů, je šance na uhádnutí přibližně 0,0015%. Referenční desktopový klient **Zfone**, implementovaný samotným autorem ZRTP, používá pro generování SAS hodnoty metodu **HMAC-SHA-256** v délce 256 bitů.

3.3.3 Metoda RS

Další z metod zabraňující útoku Man-in-the-Middle je metoda zvaná kontinuita klíče. Jedná se o metodu, kdy klíč z první relace je zapamatován za předpokladu, že nebyl indikován útok. Klíč je použit pro navázání dalších relací a metoda SAS není potřeba [17].

4 Stávající SIP řešení

Výsledkem této práce bude klient napsaný pro operační systém Android. Proto bylo rozhodnuto otestovat několik jiných klientů využívající právě protokol SIP na platformě Android. Nebylo zkoumáno chování klientů v uměle navozených podmínkách, ale klienti byli testováni v reálných podmínkách (mobilní propojení k internetu, síť s pevným připojením, za technologií NAT, v síti Wifi). Testování započalo klientem zakomponovaným přímo v systému Android a následně prošli testem SIP klienti, nabízení prostřednictvím distribučního systému Google Play.

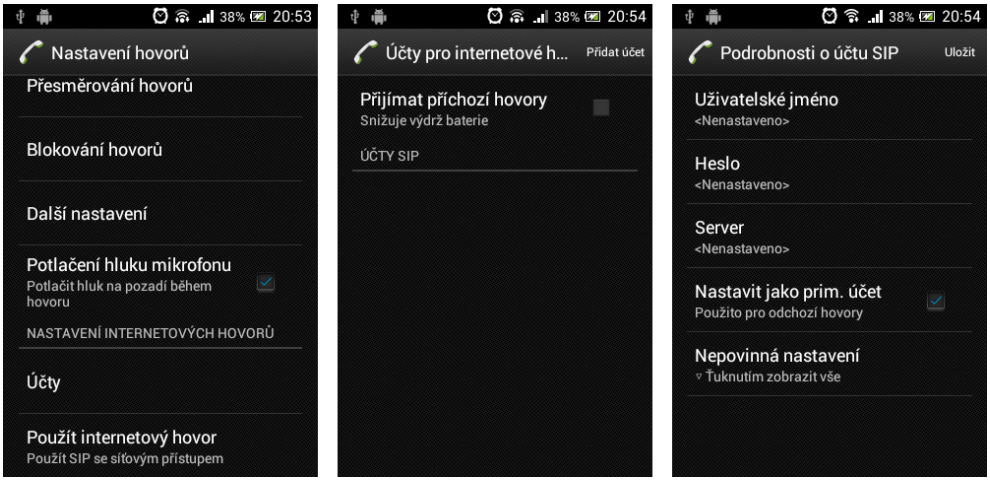
4.1 Testování klientů

Zvolení klienti byli podrobeni testu, kdy byl na dvě zařízení (kapitola 6) nainstalován téže klient. Následně byl na obou klientech přihlášen uživatelský účet ze serveru *iptel.org*. Tento server byl zvolen jako zástupce veřejných SIP registrátorů. Pro ověření požadované podpory šifrování, byl nakonfigurován a spuštěn i vlastní SIP server. Tuto funkcionalitu poskytl program Asterisk ve verzi 11.8.1. Konfigurace serveru je uvedena v příloze této práce B.

Test byl prováděn pokusem o navázání hovoru se zapnutým i vypnutým šifrováním a s různým typem připojení k internetu. V případě úspěchu pak i na zmíněném lokálním serveru.

4.1.1 Nativní klient

Systém Android obsahuje od verze 4.0 (byl zpětně přidán i do verze 2.3) jednoduchý SIP klient [19]. Nastavení tohoto klienta je možné nalézt v nastavení volání v sekci internetové hovory. Předností tohoto klienta je snadné nastavení. Mezi hlavní nevýhody se pak řadí nulová podpora jakékoliv formy šifrování.



Obrázek 9: Ukázka nativního klienta

Během testování bylo zjištěno, že některé telefony deklarovaného klienta neobsahují, i přesto, že používají verzi Androidu, která ve své čisté podobě klienta obsahuje. Dále byly objeveny i upravené verze tohoto klienta, které nebylo možné nastavit na používání mobilního internetu (fungovaly pouze přes WIFI síť). I přes zmíněnou chybějící podporu šifrování funguje integrovaný klient velmi dobře, pokud je na daném zařízení dostupný.

4.1.2 3CXPhone

Prvním zástupcem, stažitelných SIP klientů z distribučního systému Google Play, je klient s názvem 3CXPhone od společnosti 3CX. Grafické rozhranní klienta není vzhledově povedené ani přehledné. Obzvlášť nastavení klienta je velmi nekvalitně zpracované. Klient rovněž nepodporuje žádnou formu šifrování. Mezi výhody lze zařadit autory deklarovanou možnost konferenčních hovorů [20].

Obrázek 10: Ukázka klienta 3CXPhone [20]

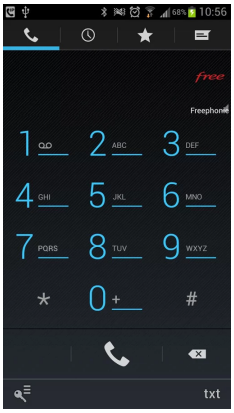
Při vyplňování uživatelského profilu je nutné zadat jméno profilu, pod kterým se bude profil v aplikaci zobrazovat a pro navázání SIP spojení nemá žádný význam. Program si jej nedokáže, alespoň volitelně, sestavit ze zadaných uživatelských údajů, což prodlužuje dobu nutnou pro nastavení klienta. Za velké negativum považuji nemožnost zadat adresu volaného účastníka jinak než čísly, stejně tak nelze volat na kontakt z jiné domény.

Klient byl podroben testu navázání hovoru, ten se však nezdařil. S ohledem na zjištěné nedokonalosti nelze tohoto klienta doporučit.

4.1.3 CSipSimple

Nejpoužívanějším stažitelný klient (podle statistik Google Play [21]) se jmenuje CSipSimple. Jedná se o svobodný klient (včetně otevřeného zdrojového kódu) se spoustou zajíma-

vých doplňkových funkcí, jako například nahrávání hovorů. Je postaven nad knihovnou PJSIP a plně využívá všech jejích vlastností [22].



Obrázek 11: Ukázka klienta CSipSimple [22]

Klient využívá možností poskytovaných systémem Android včetně vzhledu komponent a volitelné integrace do systému, což působí na uživatele velmi přívětivě. Rovněž řešení rozložení nabídek a celková přehlednost je na velmi vysoké úrovni.

Mezi hlavní výhody tohoto klienta patří [22]:

- Podpora nespočtu i HD kodeků.
- Porpora šifrování pro SIP i RTP (ZRTP).
- Porpora textových zpráv (Instant Messaging).

Klient byl podroben testu navázání hovoru, ten byl úspěšný i s jiným, prověřeným klientem. CSipSimple je velmi povedený a lze jej jen doporučit.

4.1.4 Jumblo

Klient s názvem Jumblo je nesvobodný, proprietární a ke stažení je zdarma. Při instalaci požadoval velké množství oprávnění a ve srovnání s ostatními testovanými klienty provedl vytvoření zástupce, import kontaktů a získání dalších citlivých údajů bez vědomí uživatele (při prvním spuštění se dále nedotazoval, ani neinformoval). Žádnou z těchto akcí nebylo možné v nastavení klienta zrušit [23].

Při nastavování SIP účtu je seznam podporovaných domén (poskytovatelů SIP telefonie) pevně omezen a neexistuje zde možnost přidat vlastní. Seznam poskytovatelů obsahuje pouze placené služby.

Uživatelské rozhranní je na pohled hezké, ale je neoptimalizované, zasekává se, a obsahuje chyby. S ohledem na komerční zaměření klienta bylo testování ukončeno a jeho použití nelzde doporučit.

4.1.5 Media5-fone

Klient Media5-fone je zdarma stažitelný, ale obsahuje velké množství reklam. Při prvním spuštění se bez informování uživatele zaintegroval do systému a neumožnil tuto možnost zvrátit. Pro přihlášení uživatele je nutné použít některého z pevně nabízených poskytovatelů. Těch je velké množství (není možné v seznamu hledat či filtrovat) a možnost přidat vlastního přibyla v poslední verzi. Šifrování tento klient zvládá až po dokoupení rozšíření [24].

Obrázek 12: Ukázka klienta Media5-fone [24]

Přihlášení k SIP účtu se zdařilo, hovor ale navázat nešel.

4.1.6 MizuDroid

Velkou výhodou klienta MizuDroid malá velikost aplikace a podpora starších verzí Androidu (od 2.0). Registrování účtu proběhlo bez problému, klient ale v políčku pro heslo neskrývá zadávané znaky. Klient podporuje šifrování ZRTP [25].

Test navázání hovoru mezi dvěma stejnými klienty byl neúspěšný, povedlo se ale navázat hovor s jiným klientem. Celkově tuto aplikaci nelze doporučit.

4.2 Zhodnocení

Ze všech testovaných klientů byl schopen uskutečnit registraci a následně hovor pouze CSipSimple. Ten zvládá i šifrování hovorů.

Dosažené výsledky nutně neznamenají, že by žádný z ostatních klientů nefungoval. Problémy mohly být zapříčiněny nastavením sítě, například použitím technologie NAT, což poukazuje na neúplnou implementaci některých klientů v oblasti signalizace nebo routování. Jelikož je ovšem účelem této práce vytvořit v praxi použitelného a robustního klienta, jsou s ohledem na použitelnost tyto chyby nepřípustné.

5 Návrh klienta

Jako první krok bylo nutné vybrat jazyk a technologie, v kterých bude nový klient vyvíjen. Klient musí být spustitelný na platformě Android a případně bez větších potíží portovatelný pod desktopové (Windows, Linux) či jiné systémy.

5.1 Programovací jazyk

Pro vývoj klienta se nabízejí programovací jazyky: **C**, **C++**, **Java**, anebo některý z moderních jazyků, jako je **Go**, **Vala**, **Haxe**, **Rust**.

5.1.1 Jazyk C

Jazyk C je nízkoúrovňový jazyk, vyznačující se velkou rychlostí. Doba kompilace je však u většiny kompilátorů velmi dlouhá (ve srovnání s modernějšími jazyky). Rovněž podpora novějších verzí jazyka C (C99) není, ze strany kompilátorů, příliš dobrá. S ohledem na stáří a jednoduchost jazyka zde schází mechanismy, známé z moderních jazyků (například automatická správa paměti), což by se negativně projevilo na době vývoje [26].

5.1.2 Jazyk C++

Lepší volbou by v tomto případě byl jazyk C++, nicméně jeho podpora v kompilátorech není rovněž úplná. Jazyk obsahuje velké množství syntaktických prvků, neobsahuje automatickou správu paměti a jeho kód je ve srovnání s moderními programovacími jazyky nepřehledný. Řešením by mohlo být využití jednoho z komplexních programátorských frameworků, například knihovny **Boost** [27].

5.1.3 Jazyk Java

Java je vysokoúrovňový jazyk s jednoduchou, čistou a přehlednou syntaxí. Jelikož součástí této práce je vytvořit jednoduché a přehledné API, jazyk Java je ideálním kandidátem. Je široce podporovaný v desktopových operačních systémech a díky technologii JNI je možné na některé komponenty využít rychlejšího jazyka C. Aplikace pro Android se rovněž programují v tomto jazyce. Součástí platformy Java je i sada Java collections framework a další podpůrné třídy, které velice usnadňují vývoj [28].

5.1.4 Jazyk Go

Go je nízkoúrovňový jazyk (jako C) využívající myšlenky z vyšších programovacích jazyků. Má automatickou správu paměti a velký důraz klade na bezpečnost. Bohužel, v době vzniku této práce neexistovala přímá podpora tohoto jazyka na platformě Android. Přestože by se na implementaci ideálně hodil, z výše uvedeného důvodu byl z výběru vyřazen [29].

5.1.5 Jazyk Vala

Vala vzniká jako součást projektu **Gnome** jako zdokonalená náhrada za jazyk C se syntaxí podobnou jazyku C#. Jelikož je možné ji překládat do jazyka C, její použití by bylo možné. S ohledem na pozdější vývoj práce byl tento jazyk zamítnut [30].

5.1.6 Jazyk Haxe

Haxe je vysokoúrovňový jazyk se zvláštním konceptem. Jeho zdrojový kód se nejdříve zkompiluje na jiný jazyk (C#, C++, Java, JavaScript, ...). Pro použití tohoto jazyka by bylo nutné nastavit různé kompilační procesy a připojit do něj různá potřebná API. Jelikož i jiné jazyky z mnou uvažovaných umožňují multiplatformní užití a Haxe by negeneroval optimální kód, byl proto z výběru vyřazen [31].

5.1.7 Jazyk Rust

Rust vzniká pod záštitou Mozilla Research a má nahradit jazyk C++ v jádře Gecko webového prohlížeče Mozilla Firefox. Jazyk přináší vysokoúrovňový přístup, přestože se snaží udržet na úrovni jazyka C (podobně jako jazyk Go). Jazyk obsahuje velké množství syntaktických pravidel a přináší některé zajímavé koncepty, jako například 4 druhy ukazatelů apod. Jazyk byl nakonec z výběru vyřazen, protože celkově by použití jazyka Rust nepřineslo projektu větší užitek [32].

5.2 Vývojové prostředí a nástroje

Aplikace pro platformu Android je možné psát v různých vývojových prostředích. Některá přímo podporují integraci Android Development Toolkit (ADT), v jiných je možné použít doplněk (plugin), případně kompilaci nastavit ručně [33].

Pro vývoj aplikací na operační systém Android vytváří společnost Google sadu nástrojů nazvanou souhrnně jako Android Development Toolkit. Součástí sady jsou:

- Nástroje pro kompilaci kódu.
- Nástroje pro ladění kódu.
- Nástroje pro převod java bytecode do dx.
- Emulátor systému Android s přednastavenými konfiguracemi.
- Programy pro komunikaci a ovládání připojených Android zařízení.

Součástí ale není sada Native Development Toolkit (NDK), pro vývoj částí aplikace v jazyce C nebo C++. Většina aplikací tuto funkcionalitu nepoužije, protože je určena především pro zvláštní druhy aplikací, které vyžadují nízkou komunikaci s hardwarem nebo je z nějakého důvodu není možné přepsat do jazyka Java [34]. NDK je vlastně jakousi obdobou JNI desktopové Javy pro Android.

Od počátku vývoje operačního systému Android existuje rozšíření pro IDE Eclipse, které umožňuje používat ADT přímo, a tím umožňuje snadný vývoj Android aplikací. Eclipse samotný podporuje vývoj v mnoha programovacích jazycích a pro celou řadu platforem. Za vývojem IDE stojí společnost **IBM**, která v listopadu 2001 uvolnila zdrojové kódy Eclipse 1.0 jako open source. Eclipse je napsán v jazyce Java, je postaven nad sadou komponent **SWT** a využívá modulární framework **OSGi** [35].

V květnu roku 2013 oznámila společnost Google vývoj nového IDE, určeného speciálně pro vývoj Android aplikací, pod názvem Android studio. Základ nového IDE je IntelliJ IDEA od společnosti JetBrains. Oproti Eclipse nabízí nástroj vylepšené vlastnosti [36], například:

- Kompilaci řízenou nástrojem Grandle.
- Rychlé opravy a refaktoring přizpůsobený speciálně potřebám Androidu.
- Podepisování aplikací přímo v IDE.
- Šablony a průvodci pro rychlé vytváření návrhů a komponent.
- Pokročilý editor se zvýrazňováním syntaxe umožňující drag-and-drop a visuální náhledy.
- Integrace služeb Google Cloud a Google App Engine.

5.3 Experimentální vývoj

Než byl započat samotný vývoj, bylo vytvořeno několik aplikací, testujících různé části protokolu SIP nebo práci s ním.

Zkoumána byla reálná komunikace SIP klienta se serverem. Cílem bylo napsat program, který bude schopen přijímat a odesílat základní zprávy protokolu SIP. Program bylo nutné naučit udržovat přihlášeného uživatele na serveru registrátora SIP.

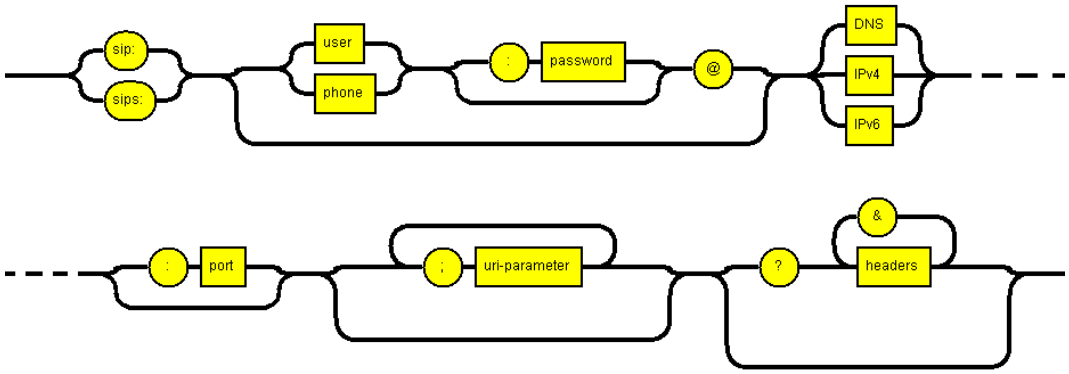
5.3.1 RailRoad diagramy

RailRoad diagramy jsou syntaktické diagramy, které byly vytvořené pro reprezentaci bezkontextové gramatiky. Jsou jednoduché, velmi přehledné a rychle pochopitelné.

K lepšímu porozumění SIP gramatice byl v rámci této práce vytvořen jednoduchý program na kreslení RailRoad diagramů. Tento pomocný program vizualizuje gramatiku zadanou pouze ve formátu JSON [38]. Bylo proto nutné ručně přepsat části gramatiky zapsané v Backus-Naurově Formě. Jedná se o syntaxi, která se používá používaná k vyjádření bezkontextové gramatiky ve standardech RFC [37].

Pro realizaci programu posloužila platforma Java a grafické prostředí komponent Swing. Program slouží pouze jako pomocný, proto má velmi omezené funkce a na výsledného klienta nemá žádný vliv. Plánuji rozšíření o možnost číst přímo zápis podle RFC 2234 [37].

Ukázka RailRoad diagramu SIP URI je zobrazena na obrázku 13.



Obrázek 13: Ukázka RailRoad diagramu SIP URI

5.3.2 Parser SIP zpráv

Platforma Android v rámci vytváření aplikací nabízí realizaci převážně v jazyce Java. Je však možné napsat některé časti programu v nativním kódu (C/C++). Tuto možnost nabízí technologie NDK, postavená na technologii JNI. Výhoda této technologie je ve využití předností jazyků C či C++, a to převážně rychlosti [39].

Vyvstala proto otázka, zda by nebylo vhodnější, aby byly SIP zprávy parsovány nativním kódem. Vznikl proto, v rámci předmětu **Operační Systémy pro Mobilní Zařízení**, experiment, jehož cílem bylo určit, zda je rychlejší parser SIP zpráv v Javě nebo C++.

Jako testovací množina posloužila zachycená registrační zpráva protokolu SIP. Ta byla opakovaně parsována nejdříve pomocí kódu v jazyce Java a následně byl stejný počet zpráv parsován kódem v C++. Součástí testovací smyčky nebyla žádná práce se sítí ani se zdroji, proto jsou naměřené hodnoty závislé pouze na době parsování zprávy. Parser napsaný v Javě zvládl zpracovat frontu zpráv přibližně za 720 milisekund. Parser v C++ zpracoval stejné množství za přibližně 1100 milisekund.

Experiment dopadl lépe ve prospěch Javy. Vše nasvědčuje tomu, že volání nativních funkcí a naopak Java funkcí v nativním kódu má velkou režii. SIP zprávy jsou moc krátké na to, aby tuto režii eliminovala rychlost parsování. Proto zvýšením množství parsovaných zpráv nezvrátí poměr výsledku experimentu.

V případě replikace výsledků jsou zdrojové kódy dostupné v příloze.

5.4 SIP frameworky

Klíčový pro vývoj vlastního klienta bylo vybrat knihovnu, která zajistí podporu protokolu SIP. Součástí výzkumné části této práce bylo nalezení, otestování a popsání následujících knihoven.

5.4.1 Vlastní řešení

Kompletní implementace celého SIP řešení není jednoduchá. Protokol SIP je neustále rozšiřován dalšími a dalšími RFC. Nejen on, ale i protokoly, které využívá, jsou neustále vyvíjeny. Spolu s tím je spojena složitost celého řešení.

Jádro samotného SIPu bylo do konce roku 2013 popsáno či jinak upraveno celkem v šesti různých RFC. U protokolu SDP skýtá tento seznam dokonce 32 různých RFC. Protokol RTP jich má pak na kontě 31.

Protokol SIP využívá některých mechanismů, které zdědil nebo si vypůjčil z jiných protokolů. Například adresy SIP jsou rozšířením adres URI, které samotné užívají adres IP nebo DNS. Také jsou využívány autentizační mechanismy, známé z jiných protokolů. Dále je v SIP využita technologie MIME. I ta je specifikována v jedenácti RFC. Celkem tedy všechna rozšíření protokolu SIP dosahují počtu 106 RFC (některá rozšíření se týkají pouze SIP serverů).

Pro vytvoření klienta, který by poskytnul všechny služby nabízené SIP technologií, by bylo nutné důkladně analyzovat a detailně nastudovat přibližně 189 vzájemně se doplňujících nebo vylučujících dokumentů.

Klíčové u protokolu SIP je porozumět základní funkcionalitě. Implementace komplexního řešení vyžaduje dlouhodobé plánování a mnoho zdrojů, což lépe zvládne větší organizovaná skupina lidí.

Z výše uvedených důvodů nebyla v rámci této práce započata implementace nového plnohodnotného SIP jádra. Součástí práce je ale několik experimentů s implementací parseru a struktury uchovávání informací protokolu SIP. Plnohodnotně je pak zapracováno API, do kterého je možné snadno napojit jádro jakéhokoliv SIP klienta [40].

5.4.2 Sofia SIP

První z testovaných knihoven je Sofia SIP postavená na SIP zásobníku, který vyvinula společnost Nokia. Je naimplementovaná v jazyce C++ a dostupná pod svobodnou licencí GNU LGPL. Primárně je určena pro cílovou platformu GNU/Linux. Knihovna splňuje podmínky dané v RFC 3261 [2, 41].

Po stažení zdrojových kódů knihovny následoval pokus o kompilaci podle dokumentace od autorů. Jako první nastal problém zastaralých souborů typu *makefile*. Při pokusu o spuštění automatické kompilace došlo k chybě, na základě které bylo zjištěno, že knihovnu není možné zkompilovat s novou verzí programu **make**. Ani po získání a instalace starší verze programu make se však kompilace nezdařila. Znovu tedy byly překontrolovány všechny požadavky a závislosti specifikované autory, přičemž žádná další chyba v postupu nebyla objevena. I přesto se kompilace knihovny nepodařila dokončit a podle nahlášených chyb ani dohledat důvod, proč tomu tak je. Z toho důvodu byla knihovna Sofia SIP vyřazena po dvou dnech pokusů práce s ní.

5.4.3 Jitsi

Původně Java klient pro desktop, který využívá stejnojmennou SIP knihovnu. Výhodou je implementace celého řešení v jazyce Java. Naopak za hlavní nevýhodu lze považovat provázanost frameworku s knihovnami AWT a Swing, které se na platformě Android nenacházejí. Po prozkoumání zdrojových kódu knihovny Jitsi, které jsou k dispozici jako open source pod svobodnou licencí GNU LGPL, bylo zjištěno, že z celkového počtu dva a půl tisíce tříd využívá knihovny AWT a Swing asi tři z nich. S ohledem na složitost celého řešení není ale jednoduché tuto závislost jednoduše eliminovat, což komplikuje port pro Android [42].

V současnosti existuje projekt (iniciován samotným autorem Jitsi), jehož snahou je zmíněný port pro systém Android vytvořit. Problém provázanosti s AWT a Swing řeší doplněním tzv. stubů požadovaných tříd se stejným názvem. Další problém, nutnost použití frameworku OSGi, řeší autoři vlastní manuální portaci této technologie do Androidu. Samotná aplikace ale není plně hotova a rovněž není stabilní. Další nevýhodou je doba kompilace. Ta je z důvodu nutného vyřazení automatické kompilace pro Android velmi zdlouhavá a při sebemenší změně kódu je nutné ji provést celou znovu. Na průměrném počítači trvá kompilace kolem pěti minut [43].

Po pěti dnech práce s touto knihovnou a snaze integrovat ji do projektu, vytvářeného v rámci této práce, byla označena jako nevhodná a bylo třeba přiklonit se k použití jiného řešení.

5.4.4 PJSIP

PJSIP je rozsáhlá knihovna napsaná v C/C++ poskytující dokonce několik rozhraní umožňujících práci s protokolem SIP. Knihovna je kompilovatelná na velkém množství platforem včetně operačního systému Android [44].

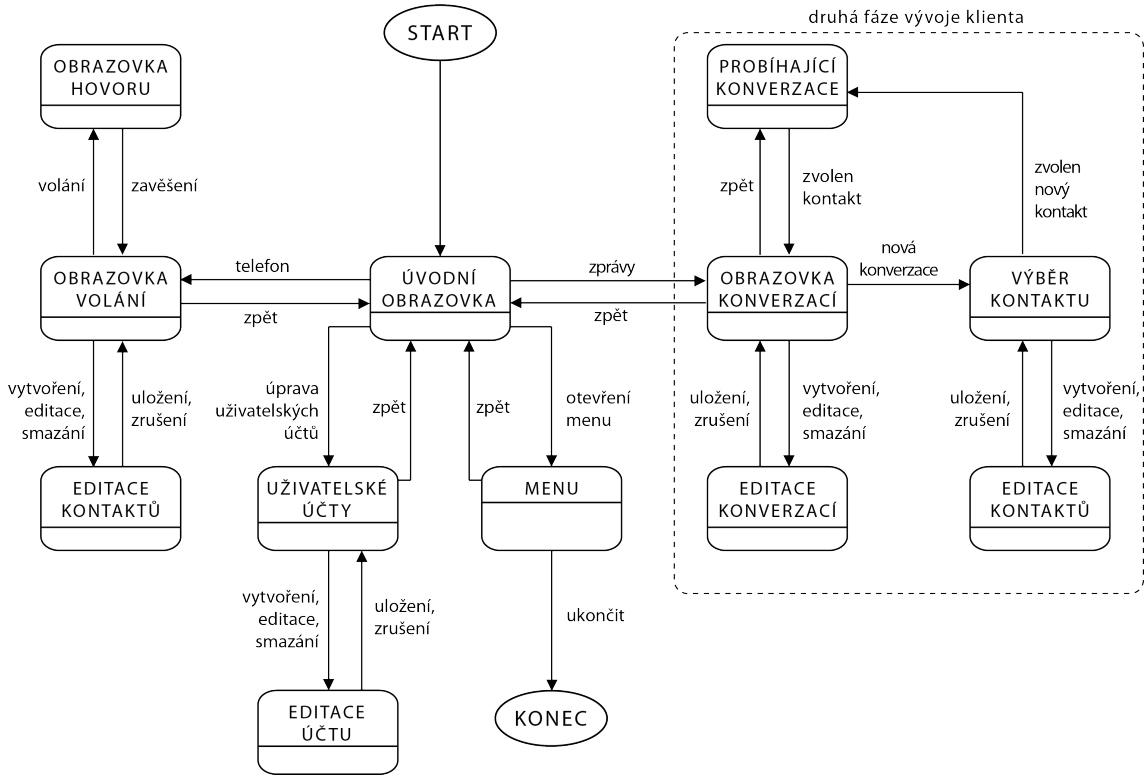
Prvním krokem bylo stažení zdrojových kódů knihovny, které pak byly dle postupu autorů zkompilovány bez větších komplikací. Zapojení do projektu se rovněž zdařilo, a proto bylo rozhodnuto tuto knihovnu použít v implementaci.

5.5 Návrh funkcionality

V případě navrhování klienta se na tento úkol pohlíželo komplexně a byly tak navrženy i věci, které bude klient obsahovat až v budoucích verzích. Z tohoto důvodu byl návrh rozdělen do dvou fází. První fáze obsahuje funkcionalitu požadovanou v rámci této práce. Konkrétními částmi jsou:

- Správa uživatelských účtů.
- Základní funkcionalita volání.
- Správu kontaktů.
- Historie událostí.

První fáze klienta obsahuje vše, co by měl komunikační klient z pohledu koncového uživatele obsahovat. Druhá fáze pak přidává podporou textových zpráv (IM). Schéma návrhu je vidět na obrázku 14.



Obrázek 14: Návrh funkcionality

5.5.1 Správa uživatelských účtů

Rozhodnutí o použití knihovny PJSIP přineslo klientu podporu víceuživatelského přístupu. Počet uživatelských účtů tedy nebude omezen, ale je nutné dát uživateli na výběr, z kterého účtu se mají provádět odchozí volání nebo jiné akce, které to vyžadují. Tento vybraný účet je označen jako **primární účet**.

Důležitým rozdílem oproti jiným SIP klientům bude možnost udržet více přihlášených uživatelských účtů současně, tedy přijímat hovory ze všech zaregistrovaných účtů uživatele.

5.5.2 Základní funkcionalita volání

Volání bude řešeno na úrovni knihovny a je tedy nutné pouze navrhnout příslušné propojení a grafickou podobu průběhu hovoru.

5.5.3 Správa kontaktů

Klient bude umožňovat uložení kontaktů, jejich správu a vyhledávání. S ohledem na jednoduchost celého řešení není v plánu implementovat plnohodnotný adresář kontaktů, jak je tomu například v operačním systému Android, ale spíše nabídnout uživateli jednoduchou a přehlednou možnost, jak zadávat složitě napsatelné a zapamatovatelné SIP adresy.

Kontakt bude reprezentován dvojicí hodnot **jméno kontaktu** a **SIP adresa**. Filtrování celého seznamu kontaktů bude probíhat souběžně se zadáváním adresy (filtrovat se bude jak podle jména, tak podle adresy) a klepnutím na vybranou položku ze seznamu kontaktů se adresa vyplní. Pokud zadávaná adresa není v seznamu, klient ji umožní jednoduše přidat doplněním jména (případně jméno odvodí z adresy).

5.5.4 Historie událostí

Poslední z funkcí, která bude součástí první fáze vývoje, je historie událostí. Událostí se v tomto případě myslí zmeškané volání, odchozí přijaté i nepřijaté hovory, případně další významné události, o kterých by měl být uživatel informován.

Součástí uchované události bude informace o tom, kdy se událost stala (případně jak dlouho trvala), obecný popis události a druhý účastník, který se na vzniku události podílel (SIP adresa volaného, apod.). Události pak budou v seznamu řazeny od nejnovějších a bude možné je smazat.

5.6 Návrh API

Přestože bude během vývoje využívána knihovna PJSIP, bude součástí práce jednotné rozhraní pro práci s protokolem SIP, umožňující integrovat libovolnou knihovnu nebo framework.

Jelikož se jedná o Android projekt psaný v programovacím jazyce Java, je nutné, aby rozhranní bylo objektově orientované. Základním stavebním kamenem tohoto rozhranní budou **interface**.

Základem je rozhranní SIP, reperezentující konkrétní implementaci protokolu SIP (jádro). V rámci životního cyklu obsahuje rozhranní metody init a free, které umožňují zavést modul a ukončit práci s ním. Jedinou další metodou tohoto rozhranní je createAccount, která na základě SIPURI objektu vytvoří a přihlásí SIP účet.

Účty protokolu SIP jsou reprezentovány rozhranním SIPAccount. Rozhranní umožňuje navázat nový hovor, odeslat zprávu (Instant message), spravovat seznam kontaktů a řídit správu událostí.

Hovor je reprezentován rozhranním SIPCall. Toto rozhranní poskytuje funkcionalitu potřebnou pro zvednutí hovoru (metoda answer). Za pomocí metody getDuration je možné zjistit dobu, po kterou hovor probíhá nebo probíhal. Metody getRemote vrací URI adresu druhého účastníka hovoru. Pokud hovor stále probíhá, vrací metoda isActive hodnotu **true**. Textovou reprezentaci stavu hovoru je možné získat pomocí metody getState.

Pro řešení reprezentace textových zpráv je zde rozhranní SIPMessage. Obsahuje čtyři metody: První z metod getFromURI umožňuje získat URI adresu odesilatele. Druhá metoda getToURI naopak vrací adresu příjemce. Tělo zprávy (getContent) je tvořeno textem, který může být různého typu. Typ těla zprávy lze získat voláním metody getContentType, která vrátí MIME typ obsahu. Výchozím typem obsahu zprávy je "text/plain".

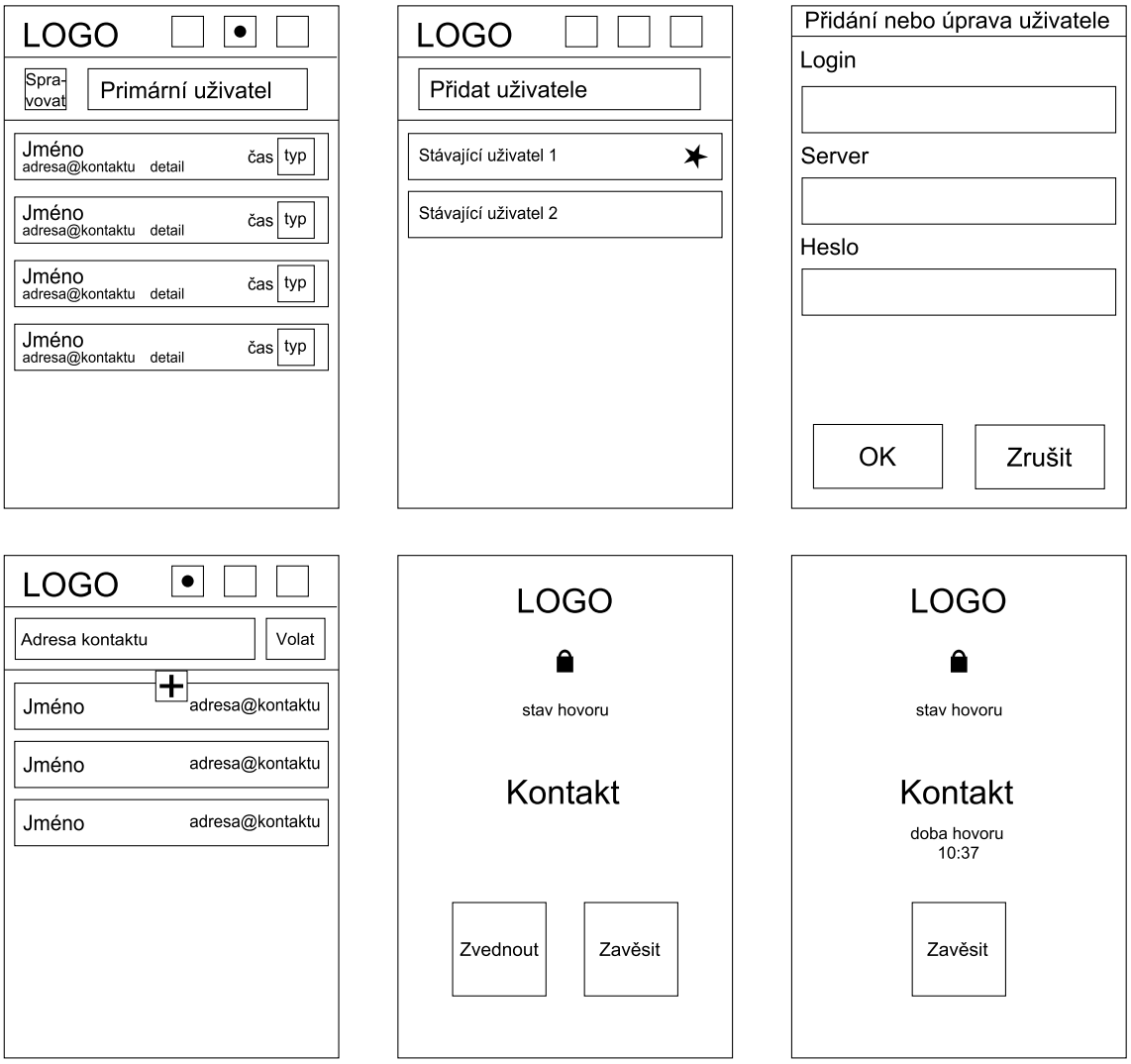
Kontakty jsou reprezentovány rozhraním SIPContact. Každý kontakt má svou přezdívku, kterou lze získat voláním metody getNick. Další částí kontaktu je jeho URI adresa, kterou lze získat voláním metody getURI.

Rozhranní SIPAccountListener reprezentuje události týkající se uživatelského účtu vyvolané ze strany serveru. Pokud dojde ke změně stavu přihlášení uživatele, vyvolá se událost onRegistrationStateChange. V případě příchozího hovoru se spustí metoda onCall, pro příchozí zprávu zase metoda onMessage.

5.7 Návrh GUI

S ohledem na navrženou funkcionalitu a způsob vytváření aplikací pro operační systém Android bylo nutné navrhnout grafické rozložení všech obrazovek (tzv. aktivit). Rozvržení musí zohledňovat nutnost dotykového ovládání, mělo by být přehledné a pro uživatele intuitivní a jednoduché.

Jako předloha pro rozvržení komponent byl použit obrázek 15.



Obrázek 15: Návrh rozložení komponent

6 Implementace

Na základě předchozích zjištění, popsaných v kapitole 5, bylo rozhodnuto projekt vyvíjet v jazyce Java za použití prostředí Eclipse. Vývoj probíhal na operačním systému Xubuntu 13.10 a k testování vyvíjené aplikace byla k dispozici dvě fyzická zařízení:

- LG P500 Optimus One [45]
  - Display: 3,2", 320 x 480, 180 DPI
  - Operační paměť: 512 MB
  - Interní paměť: 165 MB
  - Procesor: Qualcomm MSM7227 (1x ARM1136, 600 MHz, 65nm)
  - Android: 2.3.3 Gingerbread, API 10 (aktualizován)
- Sony Xperia tipo [46]
  - Display: 3,2", 320 x 480, 180 DPI
  - Operační paměť: 512 MB
  - Interní paměť: 3 GB
  - Procesor: Qualcomm MSM7225A (1x Cortex-A5, 800 MHz, 45nm)
  - Android: 4.0.4 Ice Cream Sandwich, API 14 (aktualizován)

Jako jádro klienta byl zvolen komunikační framework PJSIP (kapitola 5.4). Kvůli rozšiřitelnosti klienta a nezávislosti na zvolené implementaci protokolu SIP byla vytvořena mezivrstva, popsaná v kapitole 5.6.

Z důvodu zpětné kompatibility aplikace se systémem Android 2.0 (API 5) nebyly během vývoje využity žádné části Androidu přidané v novějším API.

6.1 Kompilace PJSIP pro Android

V první řadě bylo nutné zprovoznit PJSIP pro systém Android. PJSIP lze kompilovat pro řadu platforem včetně operačního systému Android. S ohledem na další vývoj bylo nutné sestavit prázdný projekt s podporou knihovny PJSIP. K tomu je potřeba nainstalovat následující součásti [47]:

- Zdrojové kódy knihovny PJSIP (verze 2.2.1) z adresy *www.pjsip.org/download.htm*.
- Nástroj SWIG alespoň ve verzi 2.0.5.
- Android SDK a NDK (minimálně ve verzi r8b).
- Eclipse s ADT pluginem pro jednodušší vývoj.

Jakmile je vše připraveno k použití, je třeba nastavit požadovanou kompilaci projektu zásahem do souboru **config_site.h**. Tento soubor se nachází ve složce zdrojových kódů **./pjlib/include/pj/** knihovny PJSIP (pokud zde není stačí jej vytvořit). Do souboru je nutné zapsat následující kód:

```
#define PJ_CONFIG_ANDROID 1
#include <pj/config_site_sample.h>
```

Pro zahájení kompilace pak stačí vejít do složky knihovny PJSIP a spustit následující příkazy:

```
export ANDROID_NDK_ROOT=/cesta/do/android/ndk
./configure-android
make dep && make clean && make
```

6.1.1 Šifrování

Prvním krokem k zprovoznění TLS a SSL je úspěšná kompilace knihovny OpenSSL pro systém Android. Tomuto tématu se na internetu věnuje několik projektů. Pro účely tohoto projektu byl vybrán **openssl-android**. Projekt stačí stáhnout ze serveru GitHub, rozbalit a následně spustit kompilaci pomocí nástroje ndk-build:

```
cd openssl-android
/cesta/k/ndk-build
```

Po úspěšné kompilaci je ještě zapotřebí zkopírovat soubory **libcrypto.so** a **libssl.so** ze složky **libs/armeabi/** do **lib/**.

Podporu TLS a SSL je nutné zapnout v souboru **config_site.h** následujícími definicemi:

```
#define PJ_HAS_SSL_SOCK 1
```

Příkaz potřebný pro konfiguraci kompilace je potřeba upravit přidáním parametru **withssl**, který obsahuje cestu k openssl. Výsledný příkaz tedy vypadá následovně:

```
./configure-android --with-ssl=/cesta/do/openssl-android
```

6.1.2 Zjednodušení kompilace

Pro zjednodušení a automatizaci postupu popsaného v této kapitole byl vytvořen skript pro Bash jménem **PREPARE.sh**, který stačí umístit do kořenové složky projektu a spustit. Struktura projektu je následující:

- **.classpath** - soubor generovaný Eclipse s ADT
- **.project** - soubor generovaný Eclipse s ADT

- **.settings** - soubor generovaný Eclipse s ADT
- **AndroidManifest.xml** - popisuje aplikaci včetně jejich komponent, oprávněních atd.
- **assets** - obsahuje jinam nezařaditelné zdroje (např. fonty)
- **bin** - zkompilované java soubory, zdroje a kompilát výsledné aplikace (.apk)
- **doc** - dokumentace vygenerovaná nástrojem Javadoc
- **gen** - java soubory generované Eclipse s ADT, například zdroje do R.java
- **jni** - obsahuje nativní zdrojové kódy (+ makefile) pro NDK
- **libs** - vlastní linkované knihovny
- **output** - zdrojové kódy PJSIP vygenerované pro Android skriptem PREPARE.sh
- **pjproject-2.2** - zdrojové kódy PJSIP
- **PREPARE.sh** - Bash skript generující PJSIP pro Android
- **proguard-project.txt** - soubor s nastavením pro nástroj ProGuard
- **project.properties** - obsahuje nastavení projektu (API na které je aplikace cílena)
- **README.md** - informační soubor ve formátu Markdown
- **res** - složka se zdroji pro Android
- **src** - složka se zdrojovými kódy (.java)

Na začátku skriptu jsou dvě proměnné, které je nutné korektně nastavit. Proměnná OFOUS_PJDIR odkazuje na složku se zdrojovými kódy knihovny PJSIP. Druhá proměnná, OFOUS_NDKDIR, odkazuje na složku s funkčním Android NDK.

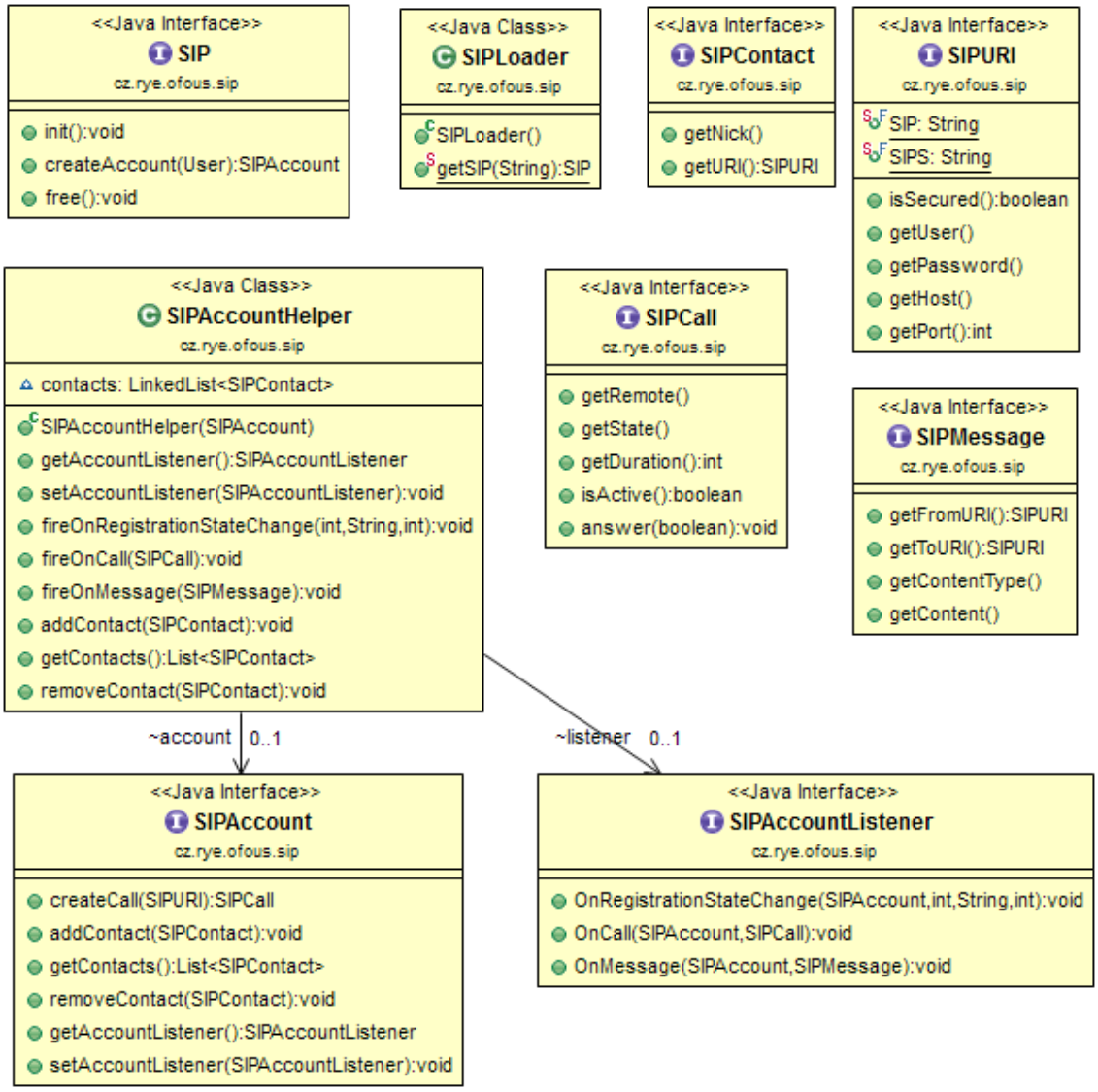
6.1.3 Zapojení PJSIP do API

Jak už bylo řečeno, klient podporuje různé implementace protokolu SIP. Aby tohoto bylo možné docílit, obsahuje klient třídu jménem SIPLoader, která umožňuje načtení jádra SIP za běhu programu. Třída SIPLoader je statická a obsahuje jedinou metodu getSIP. Parametrem je řetězec obsahující plné jméno třídy a výstupem je objekt mající rozhranní SIP.

Postup popsaný v kapitole 5.6 vytvoří automaticky sadu tříd, kterou je nutné zabalit do vlastních tříd s odpovídajícím rozhranním. Implementace PJSIP se nachází v balíčku **cz.rye.ofous.pjsip**.

Rozhranní	Implementace	Původní třída
cz.rye.ofous.sip.SIPAccount	cz.rye.ofous.pjsip.PJAccount	org.pjsip.pjsua2.Account
cz.rye.ofous.sip.SIPCall	cz.rye.ofous.pjsip.PJCall	org.pjsip.pjsua2.Call
cz.rye.ofous.sip.SIP	cz.rye.ofous.pjsip.PJSIP	java.lang.Object

Tabulka 2: Mostní implementace pro PJSIP



Obrázek 16: UML diagram API

6.2 Hlavní třída aplikace

Třída je potomkem **android.app.Application** a reprezentuje samotnou instanci klienta. Pomocí statických metod umožňuje přístup ke všem klíčovým prvkům aplikace a stará se o bezchybný chod SIP jádra (od jeho načtení po uvolnění z paměti). Vedle částí, popsaných v následujících kapitolách, umožňuje přístup k objektu SIP jádra (prostřednictvím metody getSIP()), hlavnímu úložišti klienta (metoda getSQL()), objektu probíhajícího hovoru (metoda getActiveCall()), lokalizovaným řetězcům atd. Dále je možné skrze tuto třídu navázat nový hovor, nastavit primární uživatelský účet nebo si usnadnit práci s GUI (například globální aplikace vlastního fontu pro celý klient).

6.3 Rozšíření aktivit

S ohledem na kompatibilitu se staršími verzemi systému Android, není možné využít některé moderní prvky GUI a funkcionality, známé z nových verzí systému. Jedná se konkrétně o jednotnou hlavičku aktivit nebo sloučení více aktivit do posouvatelné skupiny.

Z tohoto důvodu byla vyvinuta třída **cz.rye.ofous.SwipeActivity**, která je použita jako předek všech aktivit klienta. Třída automaticky doplní schopnost přesouvání se mezi zřetězenými aktivitami a to včetně animací. Rovněž umožňuje navázat k hlavičce aktivity všechny nutné posluchače událostí.

6.4 Správa uživatelských účtů

Pro správu uživatelských účtů je v práci sada podpůrných tříd a rozhranní. Pro zobrazování seznamu účtů se v programu nachází dvojice tříd **cz.rye.ofous.gui.UserListAdapter** a **cz.rye.ofous.gui.UserView**. Adaptér využívá návrhového vzoru *singleton* a udržuje všechna data v perzistentním stavu za použití jejich ukládání do SQL. Druhá ze tříd pak slouží jako jednoduchá komponenta reprezentující položku seznamu, zobrazující data uživatelského účtu.

Seznam uživatelských účtů je rozdělen na dvě části, z nichž první obsahuje úplný seznam a druhá pak vybranou podmnožinu tzv. oblíbených účtů, což jsou ty, které klient udržuje aktivně přihlášené.

O samotné přihlašování a odhlašování se stará již zmíněná hlavní třída aplikace, která rovněž udržuje primární uživatelský účet. Adaptér automaticky komunikuje s hlavní třídou a hlásí všechny nově přidané, upravené nebo smazané uživatelské účty, případně i změny v seznamu oblíbených účtů.

Logickou část uživatelských účtů představuje třída **cz.rye.ofous.logic.User**, která představuje inteligentní kontejner pro uchování informací o uživatelském účtu. Vedle pomocných metod umožňujících získat adresu účtu v různém tvaru apod., implementuje třída i rozhranní **android.os.Parcelable**, díky kterému je možné přenášet instanci třídy mezi aktivitami.

6.5 Volání

Funkcionalita volání je v klientovi reprezentována pomocí dvou aktivit, konkrétně **CallActivity** a **CallingActivity**. První jmenovaná slouží k tomu, aby mohl uživatel iniciovat hovor zadáním adresy volaného. Součástí aktivity je funkcionalita seznamu kontaktů, která je popsána v kapitole 6.6. Druhá potom reprezentuje obrazovku probíhajícího hovoru ať už vyvolaného ze strany aplikace nebo příchozím hovorem ze sítě. Aktivita automaticky udržuje všechny informace o hovoru a zpracovává události vykomunikované s hlavní třídou aplikace. Rovněž se stará o korektní zadávání událostí do historie.

6.6 Seznam kontaktů

Podobně jako je tomu se seznamem uživatelských účtů, i zde je použita třída představující adaptér a druhá představující položku v seznamu. Editace je v tomto případě prováděna pomocí jednoduchého okna řešeného třídou **android.app.AlertDialog**. Dalším významným rozdílem je pak aplikace rozhranní **android.widget.Filterable**, které umožňuje filtrovat položky v seznamu ihned během zadávání adresy.

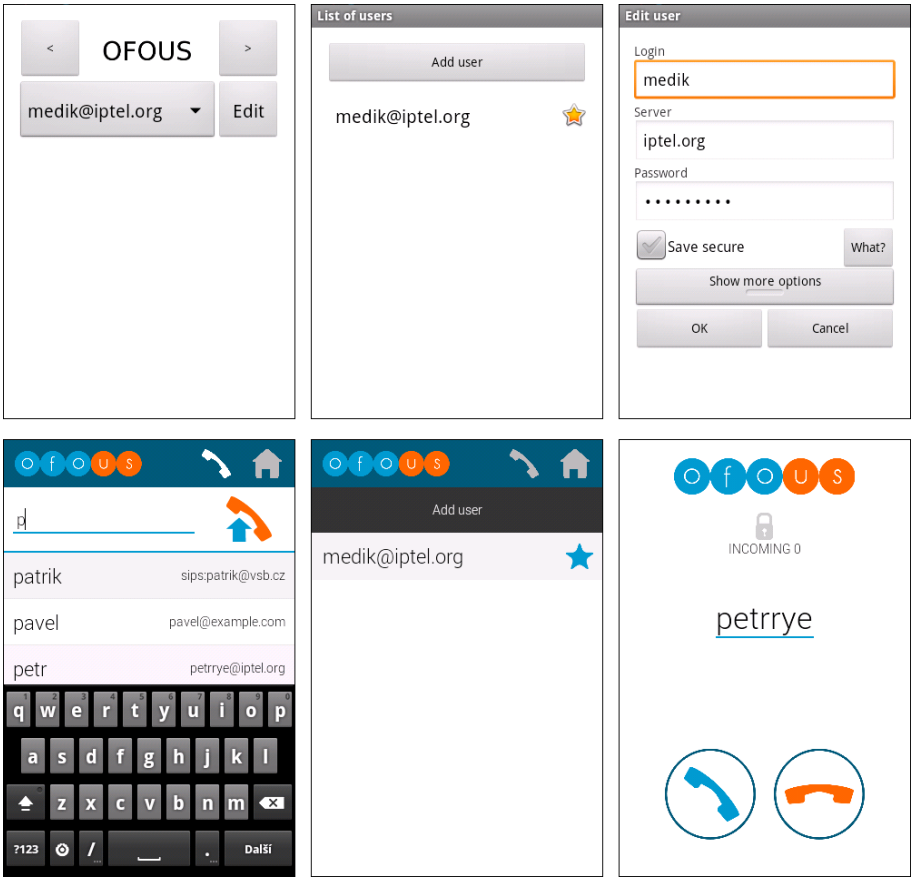
6.7 Uchovávání historie

Historie je reprezentována seznamem událostí řazeným od nejnovějších. Perzistenci opět zajišťuje SQL databáze. Události mohou být několika typů, a další rozšiřitelnost je možná díky rozhranní **cz.rye.ofous.logic.HistoryEvent**, které umožňuje získat všechna potřebná data události.

7 Optimalizace a testování

Součástí optimalizačního procesu není pouze vylepšení a urychlení funkční části aplikace, ale i zdokonalení použitelnosti aplikace z pohledu uživatele.

V rámci finálních úprav klienta byl upraven vzhled všech aktivit klienta, aby celá aplikace působila jednotně a uceleně. Tlačítka, která byla původně opatřena textem, byla v mnoha případech nahrazena ikonami a jejich původní textová informace byla použita jako pomocný popis obsahu komponenty. Rovněž všechny řetězce aplikace byly korektně umístěny do zdrojů, aby se tak usnadnila případná lokalizace celého klienta. Díky možnostem, které nabízí zdroje systému Android, je možné změnit barevnost či vzhled celé aplikace jednoduchou úpravou XML souboru, bez nutnosti zásahu do funkční části aplikace. Stav před a po grafických úpravách klienta je viditelný na obrázku 17.



Obrázek 17: Ukázka před a po aplikaci nového vzhledu

K sjednocení vzhledu přispěla i změna fontu, který je zakomponován jako zdroj přímo v aplikaci, a proto je zaručeno stejné zobrazení na všech verzích systému Android. Jelikož připojení vlastního fontu ke komponentám pomocí XML zdrojů je možné až v posledních verzích systému Android, bylo nutné vyřešit, jak zajistit aplikaci fontu pro všechny kom-

ponenty klienta. K tomuto účelu byla vytvořena metoda applyTypeface(View view), která pomocí typové konvence a reflexe jazyka Java vyvolá metodu setTypeface na všech komponentách hierarchie, které ji podporují.

Testování klienta pak probíhalo po celou dobu jeho implementace, a to už od brzkých fází vývoje. K testování funkčnosti byla využita metoda známá z extremního programování, tzv. unit testy [48]. Díky tomu byla většina klíčových částí, včetně schopnosti navázat a udržet hovor, vyvinuta a odladěna již v počátcích implementace.

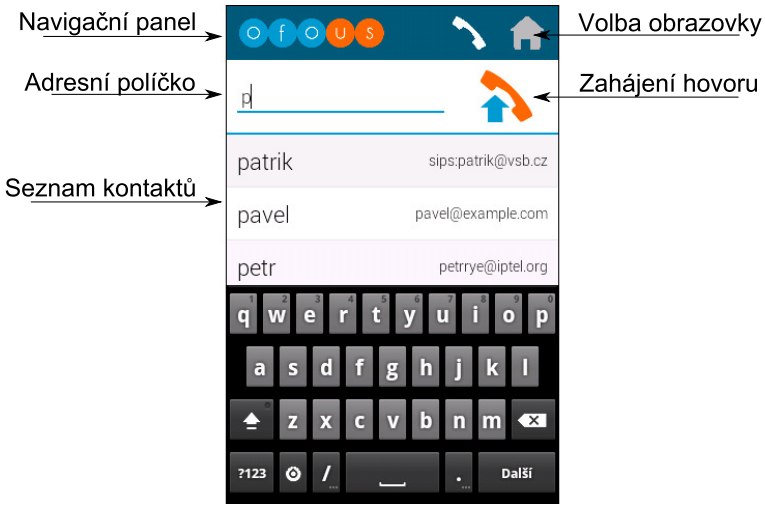
Po uzavření všech prvků, které jsou součástí první fáze vývoje, bylo provedeno komplexní otestování klienta jako funkčního celku. Navázání hovoru, ať už směrem z nebo do klienta, bylo úspěšné bez ohledu na to, zda šlo o komunikaci dvou instancí tohoto klienta nebo komunikaci s klientem CSipSimple.

8 Dokumentace

V rámci této práce byla vypracována uživatelská i programátorská dokumentace. Programátorská dokumentace byla vygenerována nástrojem Javadoc z komentářů zdrojového kódu a lze ji nalézt na přiloženém CD ve složce **doc**. Uživatelskou dokumentaci představuje dokument formátu pdf, obsahující příručku pro uživatele s návodem, jak klienta obsluhovat. Stručný výtah z tohoto dokumentu, který je opět přiložen na CD, je popsán dál v této kapitole.

8.1 Základní ovládání klienta

Při prvním spuštěním klienta se uživateli zobrazí obrazovka s prázdným seznamem uživatelských účtů a výzvou o přihlášení. Stiskem tlačítka **Add user** se objeví formulář, do kterého stačí vyplnit kontaktní údaje a po potvrzení stiskem tlačítka **OK** je uživatel uložen a přihlášen.



Obrázek 18: Rozložení obrazovek s popisem částí

Otevření obrazovky hovoru je možné stiskem ikony zobrazující sluchátko, která se nachází v záhlaví obrazovky. Klepnutím do políčka adresy (nachází se nalevo od velkého tlačítka zahajujícího hovor) je možné vyplnit adresu volaného. V případě, že se adresa nenachází v seznamu kontaktů, je možné ji stiskem tlačítka se symbolem plus do kontaktu přidat.

9 Závěr

Po důkladném seznámení se stavem na trhu dostupných VoIP klientů, s důrazem na aplikace využívající k signalizaci protokol SIP, bylo zjištěno, že 4 z 6 testovaných klientů (náhodný testovaný vzorek) jsou k praktickému použití nevhodné. Důvodem je především omezená funkcionalita, autory vynucená omezení (především seznam poskytovatelů) nebo absence jakéhokoliv šifrování či zabezpečení.

Co se týče stavu samotného protokolu SIP, tak mezi jeho největší negativa lze zařadit roztříštěnost specifikace mezi mnoho dokumentů, které se navzájem doplňují a vylučují a těžko se v nich orientuje. Samotná specifikace SIP označuje protokol za snadno implementovatelný díky své textové podobě. To usnadňuje implementaci protokolu v prvních fázích, ale po přidávání a rozšiřování funkcionality roste i složitost vznikajícího kódu. V této práci zkoumané frameworky protokolu SIP toto jen potvrzují.

K vývoji nového SIP klienta byl tedy použit již existující framework PJSIP, který, jako jeden z mála, obsahuje funkční implementaci všech, pro tuto práci potřebných, částí protokolu SIP. Pro jeho propojení do samotné aplikace vzniklo univerzální programové rozhranní, která v případě potřeby umožňuje vyměnit PJSIP za jakékoliv jiné řešení.

Vývoj byl uskutečněn v programovacím jazyce Java za použití sady nástrojů ADT a prostředí Eclipse. V první fázi vývoje vznikl funkční klient, který byl následně doplněn o vylepšení vzhledu. Díky využití PJSIP je rovněž plně podporováno šifrování, čímž je splněna podmínka bezpečnosti. Klienta je možné spustit na jakémkoliv operačním systému Android verze 2 a vyšší.

Účel práce byl splněn a výsledný klient odpovídá požadavkům i očekáváním. V budoucnu pak může být rozšířen o možnost zasílání a přijímání textových zpráv (návrh na provedení je součástí práce) a případně o další možnosti, které protokol SIP a případně framework PJSIP nabízí.

10 Reference

[1] *RFC 2543 - SIP: Session Initiation Protocol* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.ietf.org/rfc/rfc2543.txt

[2] *RFC 3261 - SIP: Session Initiation Protocol* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.ietf.org/rfc/rfc3261.txt

[3] Miroslav Vozňák: *Voice over IP*. 1. vyd. Ostrava: VŠB - TECHNICKÁ UNIVERZITA OSTRAVA, 2009, 176 s. ISBN 978-80-248-1828-3.

[4] *RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.ietf.org/rfc/rfc3986.txt

[5] *RFC 3428 - Session Initiation Protocol (SIP) Extension for Instant Messaging* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.ietf.org/rfc/rfc3428.txt

[6] *Penetrační testy v IP telefonii* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://archiv.cesnet.cz/akce/2010/vzdalena-spoluprace/p/voznak-penetracni-testy.pdf

[7] Stuart McClure, Joel Scambray, George Kurtz: *Hacking bez záhad*. 5., aktualizované a doplněné vydání: Grada Publishing, a.s. OSTRAVA, 2007, 520 s. ISBN 978-80-247-1502-5.

[8] ŘEZÁČ, Filip. *Zabezpečená komunikace na SIP protokolu* [online]. 2009 [cit. 2014-05-02]. Dostupné z: http://dspace.vsb.cz/handle/10084/74097. Diplomová práce. Vysoká škola báňská - Technická univerzita Ostrava.

[9] *Zabezpečení multimediálního přenosu v reálném čase* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://realtimesecure.asp2.cz/sip.aspx

[10] *RFC 1889 - RTP: A Transport Protocol for Real-Time Applications* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.ietf.org/rfc/rfc1889.txt

[11] *RFC 3550 - RTP: A Transport Protocol for Real-Time Applications* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.ietf.org/rfc/rfc3550.txt

[12] *Streaming media (4): transportní protokoly RTP/RTCP | DSL.cz* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.dsl.cz/clanek/60-streaming-media-4-transportni-protokoly-rtp-rtcp

[13] *File:Vzorkovani2.svg - Wikimedia Commons* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://commons.wikimedia.org/wiki/File:Vzorkovani2.svg

[14] *RFC 3711 - The Secure Real-time Transport Protocol (SRTP)* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.ietf.org/rfc/rfc3711.txt

[15] *RFC 6189 - ZRTP: Media Path Key Agreement for Unicast Secure RTP* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.ietf.org/rfc/rfc6189.txt

[16] *Zabezpečení multimediálního přenosu v reálném čase* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://realtimesecure.asp2.cz/srtp.aspx

[17] *Zabezpečení multimediálního přenosu v reálném čase* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://realtimesecure.asp2.cz/zrtp.aspx

[18] *Secure Calling Tutorial - Asterisk Project - Asterisk Project Wiki* [online]. 2014 [cit. 2014-04-29]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Secure+Calling+Tutorial

[19] *Native Android SIP Client Setup Configuration for Gingerbread or ICS* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.voipvoip.com/android/sip.html

[20] *3CXPhone for Phone System v11 - Aplikace pro Android ve službě Google Play* [online]. 2014 [cit. 2014-04-29]. Dostupné z: https://play.google.com/store/apps/details?id=com.tcx.sipphone

[21] *CSipSimple - Aplikace pro Android ve službě Google Play* [online]. 2014 [cit. 2014-04-29]. Dostupné z: https://play.google.com/store/apps/details?id=com.csipsimple

[22] *csipsimple - SIP application for Android devices* [online]. 2014 [cit. 2014-04-29]. Dostupné z: https://code.google.com/p/csipsimple/

[23] *Jumblo - Mobilní Sip volání - Aplikace pro Android ve službě Google Play* [online]. 2014 [cit. 2014-04-29]. Dostupné z: https://play.google.com/store/apps/details?id=finarea.Jumblo

[24] *Media5-fone - Mobilní Sip volání - Aplikace pro Android ve službě Google Play* [online]. 2014 [cit. 2014-04-29]. Dostupné z: https://play.google.com/store/apps/details?id=com.media5corp.m5f.Media5fone

[25] *MizuDroid - Mobilní Sip volání - Aplikace pro Android ve službě Google Play* [online]. 2014 [cit. 2014-04-29]. Dostupné z: https://play.google.com/store/apps/details?id=com.mizuvoip.mizudroid.GUI

[26] KAČMÁŘ, Dalibor. *Jazyk C*. Vyd. 1. Praha: Computer Press, 2001, xii, 185 s. ISBN 80-722-6295-5.

[27] PRATA, Stephen. *Mistrovství v C++*. 3. aktualiz. vyd. Překlad Boris Sokol. Brno: Computer Press, 2007, 1119 s. ISBN 978-80-251-1749-1.

[28] HEROUT, Pavel. *Učebnice jazyka Java*. 5., rozš. vyd. České Budějovice: Kopp, 2010, 386 s. ISBN 978-80-7232-398-2.

[29] *The Go Programming Language Specification - The Go Programming Language* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://golang.org/ref/spec

[30] *Projects/Vala/Documentation - GNOME Wiki!* [online]. 2014 [cit. 2014-04-29]. Dostupné z: https://wiki.gnome.org/Projects/Vala/Documentation

[31] *Documentation - Haxe* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://haxe.org/doc

[32] *The Rust Reference Manual* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://static.rust-lang.org/doc/master/rust.html

[33] MURPHY, Mark L. *Android 2: průvodce programováním mobilních aplikací*. Vyd. 1. Brno: Computer Press, 2011, 375 s. ISBN 978-80-251-3194-7.

[34] *Android NDK | Android Developers* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://developer.android.com/tools/sdk/ndk/index.html

[35] *Eclipse documentation* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.eclipse.org/documentation/

[36] *Getting Started with Android Studio | Android Developers* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://developer.android.com/sdk/installing/studio.html

[37] *RFC 2234 - Augmented BNF for Syntax Specifications: ABNF* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.ietf.org/rfc/rfc2234.txt

[38] *JSON* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://json.org/json-cz.html

[39] *JNI Tips | Android Developers* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://developer.android.com/training/articles/perf-jni.html

[40] *SIP Standards* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.packetizer.com/ipmc/sip/standards.html

[41] *Sofia-SIP Library* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://sofia-sip.sourceforge.net/

[42] *jitsi.org | Jitsi* [online]. 2014 [cit. 2014-04-29]. Dostupné z: https://jitsi.org/

[43] *jitsi/jitsi-android - GitHub* [online]. 2014 [cit. 2014-04-29]. Dostupné z: https://github.com/jitsi/jitsi-android

[44] *PJSIP - Open Source SIP, Media, and NAT Traversal Library* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.pjsip.org/

[45] *LG P500 - Android - Mobilní telefony - LG Electronics CZ* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://www.lg.com/cz/telefony/lg-P500-optimus-one

[46] *Xperia^{TM} tipo, Navy Blue - Discontinued Android Smartphone Sony Store - Sony US* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://store.sony.com/xperia-tipo-navy-blue-zid27-XTST21SSBLU/cat-27-catid-EOL-Android-Smartphone

[47] *Getting-Started/Android – pjsip Open source SIP, media, and NAT traversal stacks/libraries for smartphones* [online]. 2014 [cit. 2014-04-29]. Dostupné z: https://trac.pjsip.org/repos/wiki/Getting-Started/Android

[48] Kent Beck, Martin Fowler: *Planning Extreme Programming*. 1. vydání: Addison Wesley 2000, 160 s. ISBN 0-201-71091-9.

A Zdrojové kódy a výsledný klient

Na přiloženém CD se nachází spustitelný klient spolu se zdrojovými kódy. Dále jsou na CD i v práci zmíněné experimentální části, včetně návodů, jak s nimi pracovat. Přiložena je i uživatelská a programátorská dokumentace vzniklého klienta.

Obsah CD

- **Diplomová práce.pdf** - samotná práce v elektronické podobě
- **doc/** - programátorská dokumentace
- **Ofous.apk** - zkompilovaný klient
- **dp/** - zdrojové kódy této práce včetně obrázků
- **src/** - obsahuje zdrojové kódy
- **src/experiment/** - obsahuje zdrojové kódy experimentu parseru
- **src/experiment/sip/** - zdrojové kódy SIP parseru
- **src/experiment/sip_parser/** - zdrojové kódy nativního SIP parseru
- **src/klient/** - zdrojové kódy klienta
- **src/openssl-android-master/** - zdrojové kódy knihovny OpenSSL
- **src/railroad_diagramy/** - zdrojové kódy
- **userdoc/** - zdrojové kódy uživatelské dokumentace
- **Uživatelská dokumentace.pdf** - uživatelská příručka

B Spuštění testovacího SIP serveru s podporou šifrování

Server byl spuštěn na operačním systému **Xubuntu 13.10**. Instalace potřebných balíků byla provedena programem **Synaptic**:

- asterisk (verze 11.8.1)
- asterisk-config (verze 11.8.1)
- asterisk-modules (verze 11.8.1)
- srtp-utils (1.4.5)
- libsrtp0 (1.4.5)

Certifikáty

Po instalaci bylo nutné stáhnout zdrojové kódy programu Asterisk (z adresy: *http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-11-current.tar.gz*) a za pomocí skriptu **ast_tls_cert**, umístěného ve složce **contrib/scripts**, byly vytvořeny potřebné certifikáty.

Serverový certifikát (spolu s certifikační autoritou) podepsaný sám sebou (self-signed), byl vytvořen příkazem:

```
./ast_tls_cert -C 178.72.229.7 -O "sip.rye.cz" -d /etc/asterisk/keys
```

Klientský certifikát pro aplikaci klienta, který byl spuštěn na telefonech LG a SONY, byl vygenerován pomocí příkazů:

```
./ast_tls_cert -m client -c /etc/asterisk/keys/ca.crt
  -k /etc/asterisk/keys/ca.key -C lg.sip.rye.cz
  -O "sip.rye.cz" -d /etc/asterisk/keys -o lg

./ast_tls_cert -m client -c /etc/asterisk/keys/ca.crt
  -k /etc/asterisk/keys/ca.key -C sony.sip.rye.cz
  -O "sip.rye.cz" -d /etc/asterisk/keys -o sony
```

Výsledné certifikáty **lg.pem**, **sony.pem** a **ca.crt** jsou určeny pro samotné klienty.

Konfigurace

Samotná konfigurace SIP serveru v souboru **/etc/asterisk/sip.conf** byla provedena následovně [18]:

```
tlsenable=yes
tlsbindaddr=0.0.0.0
tlscertfile=/etc/asterisk/keys/asterisk.pem
tlscafile=/etc/asterisk/keys/ca.crt
```

```
tlscipher=ALL
tlsclientmethod=tlsv1

;lg
[1001]
type=peer
secret=heslo
host=dynamic
dtmfmode=rfc2833
disallow=all
allow=g722
allow=ulaw
transport=tls
srtpcapable=yes

;sony
[1002]
type=peer
secret=heslo
host=dynamic
dtmfmode=rfc2833
disallow=all
allow=g722
allow=ulaw
transport=tls
srtpcapable=yes
```